LEARNING FROM PAIRWISE SIMILARITY FOR VISUAL CATEGORIZATION

A Dissertation
Presented to
The Academic Faculty

By

Yen-Chang Hsu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2020

LEARNING FROM PAIRWISE SIMILARITY FOR VISUAL CATEGORIZATION

Approved by:

Dr. Zsolt Kira, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Patricio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Dhruv Batra
School of Interactive Computing
Georgia Institute of Technology

Dr. Judy Hoffman
School of Interactive Computing
Georgia Institute of Technology

Dr. Phillip Odom
Aero Trans Adv Sys (ATAS)
Georgia Tech Research Institute

Date Approved: March 23, 2020

To my family.

ACKNOWLEDGEMENTS

I want to express my most profound appreciation to my fantastic advisor, Zsolt Kira. He has been an invaluable source of insight, ideas, and practical guidance through each stage of the process. He has an amiable and calm demeanor, creating an delightful environment for me to explore many interesting research problems. He consistently put in the time and energy through both successes and failures. It is my fortune to work with him, and I am proud of the work we have done together over the years. I am also extremely thankful to my thesis committee members for their time and critique of this work. Their insightful feedback improved the quality of this thesis.

I want to extend my deepest gratitude to Yilin Shen, Hongxia Jin, and Jiawei Huang. I enjoyed the internships at Samsung Research America and Honda Research Institute under their mentorship. The research done during the internships inspires several chapters of this thesis.

I have been very fortunate to work with a wonderful set of collaborators: Zhaoyang Lv, Phillip Odom, Joel Schlosser, Zheng Xu, Yen-Cheng Liu, Anita Ramasamy, Jonathan Balloch, Junjiao Tian, and James Smith. Their support and contributions give me the courage to battle every submission deadline. Many thanks go to my friends and lab colleagues at Georgia Tech. They always kindly give me suggestions and feedback, sharing their experience to help me gain confidence in facing challenges of research and life. Special thanks to Chuan-Yi Tang, Karen Hsu, Tyng-Luh Liu, and Hwann-Tzong Chen for their encouragement and assistance on starting this incredible journey.

Finally, none of this would have been possible without the support of my family. I would not be in this position without my parents' long-lasting support and help during my life. I cannot begin to express my thanks to my wife, Chia-Hsuan. She has been my most stalwart supporter in every endeavor and has left no stone unturned in supporting me. This thesis is dedicated to my family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Learning high-capacity machine learning models for perception, especially for high-dimensional inputs such as in computer vision, requires a large amount of human-annotated data. Many efforts have been made to construct such large-scale, annotated datasets. However, there are not many options for transferring knowledge from those datasets to other tasks with different categories, limiting the value of these efforts. While one common option for transfer is reusing a learned feature representation, other options for reusing supervision across tasks are generally not considered due to the tight association between labels and tasks. This thesis proposes to use an intermediate form of supervision, pairwise similarity, for enabling the transferability of supervision across different categorization tasks that have different sets of classes. We show that pairwise similarity, defined as whether two pieces of data have the same semantic meaning or not, is sufficient as the primary supervision for learning categorization problems such as clustering and classification.

We investigate this idea by answering two transfer learning questions: how and when to transfer. We develop two loss functions for answering how to transfer and show the same framework can support supervised, unsupervised, and semi-supervised learning paradigms, demonstrating better performance over previous methods. This result makes discovering unseen categories in unlabeled data possible by transferring a learned pairwise similarity prediction function. Additionally, we provide a decomposed confidence strategy for answering when to transfer, achieving state-of-the-art results on out-of-distribution data detection. Lastly, we apply our loss function to the application of instance segmentation, demonstrating the scalability of our method in utilizing pairwise similarity within a real-world problem.

CHAPTER 1
INTRODUCTION

The field of machine perception has made tremendous progress in image classification, object detection, semantic segmentation, and many other problems. These improvements have been driven by two ingredients: complex machine learning models and large human-annotated data. The models usually adopt deep neural networks due to their scalable capacity in learning to convert image pixels to a target. Such capacity comes with the price of millions of parameters, leading to the need of big labeled datasets to generalize, which are often associated with the problem of overfitting. The difficulty of learning such powerful models thereby depends highly on the cost of collecting the labels, which depends on the amount of human supervision involved.

In supervised learning, the human-annotated labels used in common computer vision applications are expensive. The class label takes a person 1 second per object, while an object bounding box takes 10 seconds and an object segmentation mask requires 79 seconds [1]. Those costs become astounding when the datasets are at the scale of millions of images. Furthermore, all the above labels require a-priori knowledge of all classes, and limits the form of supervision that can be leveraged. For example, the classes may be ambiguous or non-expert human annotators may be able to more easily provide information about whether two instances are of the same class or not, rather than identifying the specific class (See Figure 1.1). More importantly, learning with class-specific labels limit the classifier from being applied to unseen classes.

In contrast to human-provided supervision, some strategies create dense supervision without humans. Such strategies, called self-supervised or unsupervised learning, automatically create labels from varied natural cues, including temporal constraints, spatial constraints, self-reconstruction, or using clustering methods based on assumptions of the

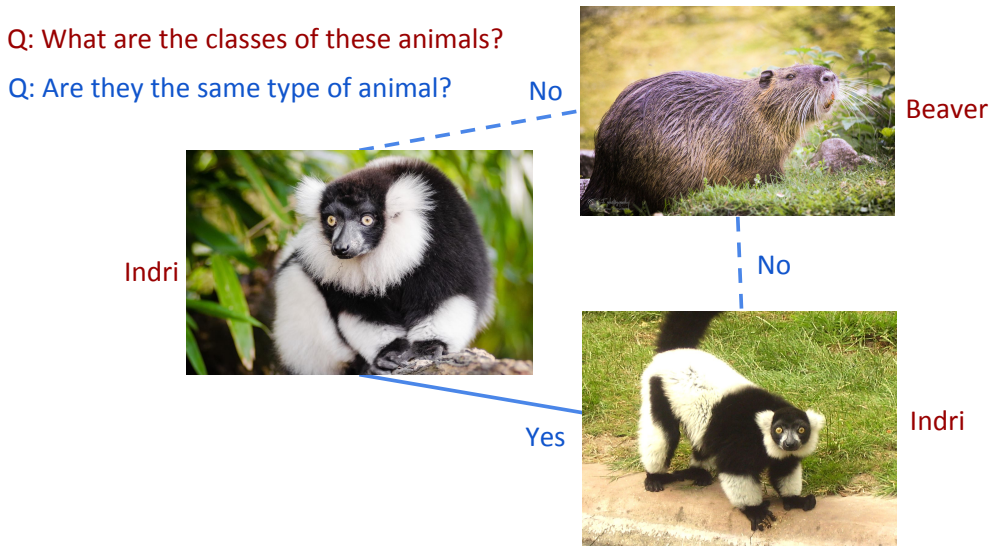


Figure 1.1: It is easier for a human to judge whether a pair of images is from the same category than labeling their classes. Images with public domain licence credit to Mathias Appel (left), Silviu Firulete (top), and Hengatie (bottom).

data distribution [2, 3]. Some of these strategies define proxy tasks, such as image colorization [4, 5], image rotation prediction [6], and image patch placement [7, 8], in which labels are obtained for free. However, models learned with these proxy tasks are only used for parameter initialization, and still require human-annotated labels for performing an actual target task. This is because the outputs of a proxy task are not aligned with the target applications, such as the class of images.

Unlike learning with a crafted proxy task, we humans can directly identify semantically coherent clusters even for unseen classes by leveraging a-priori knowledge of how categories are defined (see Figure 1.2). In other words, discovering categories should be a process which is not only based on self-supervised or unsupervised learning but also supervised learning if such human supervision is available (*e.g.* for learning how to compare a pair of images). Can a machine achieve a similar capability to discover new categories while incorporating supervision when available? In this thesis, we investigate this process as a transfer learning problem, casting the problem as questions of **what , how, and when to transfer**.

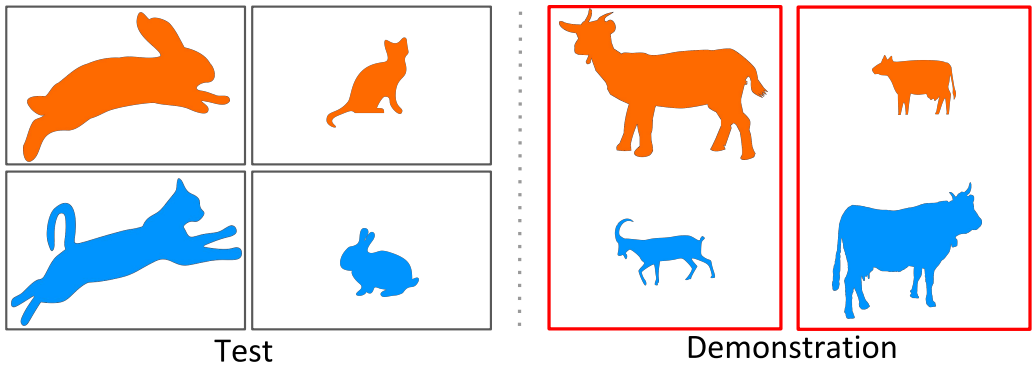


Figure 1.2: Learning to cluster: Consider grouping the four *Test* images into two clusters. There are four possible criteria: color, pose, species, and size, and the general task of grouping these items does not specify which to use. However, once the demonstration is given, the criteria for clustering can be learnt. In this case, it is species.

Regarding the question of *what to transfer*, one desideratum inspired from self-supervised learning is formulating the form of supervision in a way so that it can be generated automatically, either from a learned function or from natural cues such as spatial or temporal relationships. The form of supervision should not be class-specific, while still encoding the semantic meaning of categories. As discussed in the above paragraphs, neither the class-specific labels nor the free labels of proxy tasks can meet the above desiderata. Therefore we focus on **pairwise similarity**, which does satisfies them. We regard pairwise similarity as information indicating whether two instances (*e.g.* images for classification or pixels for segmentation) belong to the same class or not (*e.g.* the blue lines in Figure 1.1). When human supervision is available, such as a labeled dataset, pairwise similarity can be learned through a function which takes a pair of data and predicts a similarity. Then this function is what we transfer to another categorization task, which may or may not contain unseen categories, to provide free supervision for learning with the unlabeled data.

The second question is *how to transfer*. Since pairwise similarity is provided, formulating it into the optimization objective of clustering is the first choice. This strategy is closely related to constrained clustering algorithms, which use a binarized form of similarity as the constraints to construct clusters. In later sections, we will see that the existing constrained clustering methods are sensitive to noisy similarity estimates and the number

of clusters. Therefore one of the goals of this thesis is to provide robust clustering methods which enable a transferring scheme by leveraging the success of deep learning.

Lastly, *when to transfer* is a problem of determining when the learned knowledge is useful for understanding unknowns (unlabeled data). For example, a learned pairwise similarity function may make satisfiable predictions only when the unknowns are similar in a certain way to what it previously learned. In the case that the unknowns are not related the knowns (training data), the machine should neither make predictions nor discover categories, but switch to a fallback solution such as asking for human supervision. Determining which case is applicable requires a method to quantify similarities between knowns and unknowns. Therefore we address this problem by developing an out-of-distribution data detection strategy.

1.1 Thesis Statement

The pairwise similarity can be used as the primary supervision for image categorization tasks such as clustering and classification, making transferring supervisions across tasks a viable learning option.

1.2 Outline

After reviewing the related work in Chapter 2, we explore a strategy for a classical clustering problem which utilizes pairwise similarity in Chapter 3: the constrained clustering problem. In this case, the constraints are binarized pairwise similarity for indicating whether two data are from the same class or not. While most of the constrained clustering algorithms use pairwise similarity separately in metric learning and cluster assignment, we develop a new strategy to do both simultaneously in a neural network model with a single optimization objective. We further demonstrate how the proposed method enhances clustering performance and robustness against noisy similarities and an unknown number of clusters.

Chapter 4 provides a strategy to collect pairwise similarity without labels in the target domain (and therefore no extra labeling cost) via transferring a learned similarity prediction function, showing that we can discover unseen categories for images with only predicted pairwise similarities. Transfer learning is performed across domains and tasks, formulating it as a problem of learning to cluster with the optimization objectives proposed in Chapter 3.

Chapter 5 is another extension of Chapter 3, but from the view of classification. This chapter presents a new strategy for multi-class classification that requires no class-specific labels, but instead leverages pairwise similarity between data. The proposed method, meta classification learning, optimizes a binary classifier for pairwise similarities and through this process learns a multi-class classifier as a sub-module. We formulate this approach, present a probabilistic graphical model for it, and derive a surprisingly simple objective function that can be used to learn discriminative classifiers with neural network-based models.

Chapters 6 and 7 address the question of *when to transfer*. We begin with investigating the correlation between domain similarity and transferability in Chapter 6. Then, we relax the limitation of the domain similarity method used in Chapter 6, proposing two out-of-distribution data detection strategies in Chapter 7. We specifically propose to decompose confidence scoring as well as a modified input pre-processing method, showing that both of these significantly help in detecting when data comes from a different domain (distribution).

Finally, in Chapter 8, we extend the data type from image-wise similarity to pixel-wise similarity, demonstrating the scalability (*e.g.* millions of pixels) and generalizability (real-world application) of our strategy proposed in Chapter 3. Specifically, we address two applications. The first is road lane detection for an autonomous driving scenario, in which our clustering criteria can be straightforwardly applied. The second is the generic instance segmentation problem, which is challenging because the images have a much larger

amount of instances than the first case. We address such a challenge by augmenting our clustering criteria with the map coloring concept since two-dimensional images constrain the connectivity of pixels.

1.3 Contributions

This dissertation provides new methods and theories to utilize pairwise similarity for visual categorization problems. Specifically, we made the following key contributions:

- Novel clustering optimization criteria: We propose KLD-based contrastive loss (KCL), which jointly optimizes the clustering criteria while learning the feature representation with neural networks. The clustering method shows a strong robustness against the number of clusters specified to the algorithm. (Chapter 3) [9]

- Novel classification learning criteria: Our Meta Classification Likelihood (MCL) has both theoretical and empirical supports that multi-class classifiers can be learned with only pairwise similarity. (Chapter 5) [10]

- New transfer learning scheme: We demonstrate how to transfer pairwise supervision across tasks of different sets of classes, allowing new categories to be discovered. (Chapter 4) [11]

- New perspectives for out-of-distribution detection problem: We introduce the closed-world assumption to re-interpret the class posterior probability output from a discriminative classifier, inspiring a set of classifier designs which detect out-of-distribution data with a significantly improved performance. Besides, we provide new insight by dividing the problem into two cases, semantic and non-semantic shifts, pointing out the challenge of detecting unseen classes. (Chapter 7) [12]

- Demonstrated scalability and generalizability: Our KCL is applied to instance segmentation for clustering millions of pixels, demonstrating competitive results in real-

world applications. (Chapter 8) [13]

The demo code of this thesis is publicly available at https://github.com/GT-RIPL/L2C

CHAPTER 2
RELATED WORK

Constrained clustering and transfer learning are the two core concepts that we leverage for learning with pairwise similarity. We discuss them here at a higher-level while leaving each chapter to provide specific background or literature survey for situating varied sub-problems covered in the thesis.

2.1 Constrained Clustering

Constrained clustering algorithms can be categorized by how they utilize pairwise constraints (*i.e*. binarized pairwise similarity). The first set of work uses the constraints to learn a distance metric. For example, DML [14], ITML [15], SKMS [16], SKKm [16, 17], and SKLR [17]. This group of approaches closely relates to metric learning and needs a clustering algorithm such as K-means in a separate stage to obtain cluster assignments. The second group of work uses constraints to formulate a clustering loss, so that clustering results incur minimal violation against the constraints. For example, COP-KMeans [18], CSP [19] and COSC [20]. The third group uses constraints for both metric learning and the clustering objective, such as MPCKMeans [21] and CECM [22]. The fourth group does not use constraints at all. Generic clustering algorithms such as K-means [23], LSC [24], and LPNMF [25] all belong to this category. There is a long list of associated works which are summarized in survey papers, e.g. [26] and [27]. Our proposed clustering strategy belongs to the third group. Additionally, the constrained clustering methods above usually focus on semi-supervised setting where the ground-truth constraints are sparsely available. In our unsupervised setting, the ground-truth is unavailable but predicted constraints are densely available. This thesis includes all four groups of algorithms in the comparison and show the advantages of the third group.

2.2 Transfer Learning

Transfer learning aims to leverage knowledge from the source domain to help learn in the target domain, while only focusing on the performance on the target domain. Transfer learning methods make use of the knowledge gained while learning one problem and applying it to a different but related problem. The survey by Pan *et al*. [28] has systematic notions for transfer learning problems. The problems can be categorized by the distributional difference between the source and target inputs, or between the source and target outputs (label spaces). These notions lead to two settings in the thesis. The first is the cross-domain transferring where the source and target have the same output label space, but the input distributions are different. The second is the cross-task transferring, which has differences in both input distributions and output spaces. Chapter 4 has more discussions in these two settings.

In the survey [28], the author raised three research questions for transfer learning: *what, how, and when* to transfer. Most of the literature addresses the first two questions. For example, the learned parameter or weight of a model is *what* to transfer, while fine-tuning the model with only the target dataset is *how* it can be transferred. This strategy had been adopted widely by the computer vision community on varied tasks such as classification [29, 30], object detection [31], semantic segmentation [32], and image captioning [33]. This strategy does not require the source dataset to be accessible during the learning of the target task, but it requires the labels of the target task to be available. Another widely investigated problem setting is unsupervised domain adaptation, which usually assumes that the source and target data are from the same set of classes, and both datasets have to be accessible during learning, except that the labels of target data are not available. In this setting, the source data (or their representations) are *what* to transfer, while learning feature representations with minimized domain discrepancy between the source and target is *how* it can be transferred. There is a huge body of works addressing this problem [34, 35, 36,

37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51].

In this thesis, pairwise similarity is *what* we transfer, while the constrained clustering is *how* we transfer. Our combined strategy frees the learning of the target task from requiring the source data and target task labels. Note that the information transferred and methods proposed are orthogonal to other transfer mechanisms; therefore, our work can easily combine other transfer learning methods mentioned above, such as initializing the model with pre-training or adding domain adaptation criteria, discussed in Chapter 4.

The research question of *when* to transfer is rarely treated as a standalone problem. It is usually a part of a method for selecting a subset of source instances to be trained together with the target data [52, 53, 54]. This question is also addressed in the variants of unsupervised domain adaptation settings, such as open set domain adaptation [55, 56] and universal domain adaptation [57]. They assume the set of classes between the source and target partially overlap, where some unlabeled target data may not belong to any known classes. These works design methods to exclude the target data of unknown classes, treating those data as outliers. Such an idea is closely related to the problem of out-of-distribution detection [58, 59], which has a similar aim of determining *when* we can trust the predictions of a classifier deployed in an open world [60, 61]. Therefore, we investigate *when* to transfer through a proxy setting, the out-of-distribution data detection as a standalone problem. It allows us to design strategies independent from the transfer learning method, setting the stage for understanding how well (or poorly) this problem can be addressed.

CHAPTER 3
FROM PAIRS TO CLUSTERS

3.1 Introduction

Performing end-to-end training and testing using deep neural networks to solve various tasks has become a dominant approach across many fields due to its performance, efficiency, and simplicity. Success across a diverse set of tasks has been achieved in this manner, including classification of pixel-level information into high level categories [62], pixel-level labeling for image segmentation [63, 64], robot arm control [65], speech recognition [66], playing Atari games [67] and Go [68, 69]. All of the above techniques largely avoid sophisticated pipeline implementations and human-in-the-loop tuning by adopting the concept of training the networks to learn the target problem directly.

Clustering, a classical machine learning problem, has not yet been fully explored in a similar manner. Although some two-stage approaches have tried to learn the feature embedding specifically for clustering [14, 15], they still require using other clustering algorithms such as K-means to determine the actual clusters at the second step. Specifically, the first stage of previous works usually assume how the data is distributed in the projected space using human-chosen criteria such as self-reconstruction, local relationship preservation, sparsity [70, 71, 72, 73, 74, 75], fitting predefined distributions [76], or strengthening of neighborhood relationships [77, 3] to learn the feature embedding. Furthermore, all of these techniques then use a metric, such as Euclidean or cosine distance, in the second stage. This further introduces human-induced bias via strong assumptions, and the chosen metric may not necessarily be appropriate for the embedded space.

In this chapter, we present a framework [9], which minimizes such assumptions by training a neural network that can directly assign the clusters at the output layer. We

specifically use weak labels, in the form of *pairwise constraints* or similar/dissimilar pairs, to learn the feature space as well as output a clustering. In order to adopt the raw data and weak labels for end-to-end clustering, we present the novel concept of constructing the cost function in a manner that incorporates contrastive Kullback-Leibler divergence to minimize the statistical distance between predicted cluster probabilities for similar pairs, while maximizing the distance for dissimilar pairs. In the later section of this chapter, we will show that the framework is extremely easy to realize by rearranging existing functional blocks of deep neural networks, so it has large flexibility to adopt new layer types, network architectures, or optimization strategies for the purpose of clustering.

One significant property of the proposed end-to-end clustering is that there are no cluster centers explicitly represented. This largely differs from all of the works mentioned above. Without the centers, no explicit distance metrics need to be involved for deciding the cluster assignment. The learning of the cluster assignments is purely data-driven and is implicitly handled by the parameters and the non-linear operations of the network. Of course the outputs of the last hidden layer could be regarded as the learned features, however it is not necessary to interpret it using predefined metrics such as Euclidean or cosine distance. The networks will find the best way to utilize the embedded feature space during the same training process in order to perform clustering as well. The experimental sections will demonstrate this property, in addition to strong robustness when the number of output clusters is varied. In such cases, the network tends to output a clustering that only utilizes the same number of nodes as there are clusters intrinsically in the data.

3.2 Background

A common strategy to utilize pairwise relationship with neural networks is the Siamese architecture [78]. The concept had been widely applied to various computer vision topics, such as similarity metric learning [79], dimensionality reduction [80], semi-supervised embedding [81], and some applications to image data, such as in learning to match patches

[82, 83] and feature points [84]. The work of [85] uses the coherent nature of video as a way to collect the pairwise relationships and learn its features with a Siamese architecture. A similar idea of leveraging temporal data is also presented in the report of [86]. In addition, triplet networks, which could be regarded as an extension of Siamese, gained significant success in the application of learning fine-grained image similarity [87] and face recognition [88]. Despite the wide applicability of the Siamese architecture, to the best of our knowledge there is no report exploring them from the clustering perspective. Furthermore, while some works try to maximize the information in a training batch by carefully sampling the pair [82] or by formulating it as a triplet [87], there is no work showing how to use dense pairwise information directly and efficiently.

Our proposed implementation strategy can efficiently utilize any amount of pairwise constraints from a dataset to train a neural network to perform clustering. When the full set of constraints is given, it can compare to the vanilla networks trained using supervised classification. If only partial pairwise constraints are available, the problem is similar to semi-supervised clustering. There is a long list of previous work related to the problem. For example, COP-Kmeans [18] forced the clusters to comply with constraints and [89] added terms in spectral clustering to penalize the violation of constraints. The more closely related works perform metric learning [90] or feature re-weighting [91] during the clustering process. The recent approaches TVClust and RDP-means [92] address the problem with probabilistic models. None of these approaches, however, jointly learn the feature space in addition to clustering with neural networks.

3.3 Neural Network-based Clustering using Pairwise Constraints

In order to inject the concept of clusters into a neural network formulation, we first consider the vanilla multilayer perceptron (MLP) used for classification tasks: Each output node is associated with predefined labels and the optimization minimizes a cost function, such as cross entropy, that compares the output labels (or the distribution over the labels) provided

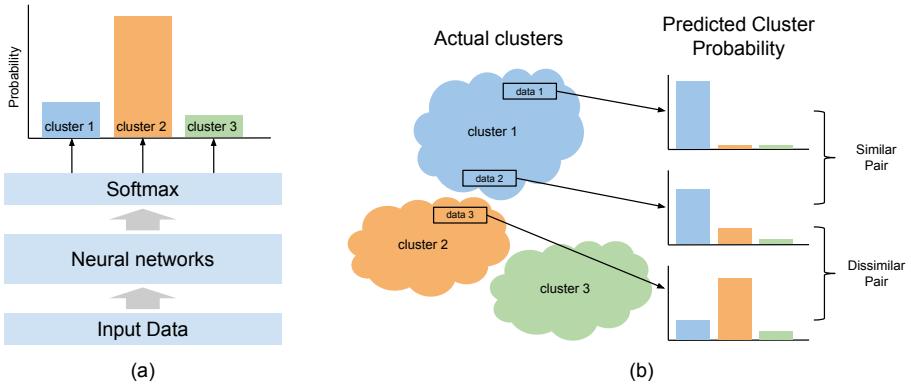


Figure 3.1: Illustration of (a) how neural networks output the distribution of possible clusters given a sample, (b) the example of predicted cluster distribution between similar/dissimilar pairs.

by the network for a set of instances and the corresponding ground truth labels. We start from this model and remove the hard association between labels and network outputs. The idea is to only use pairwise information and define the output nodes in a manner such that they can represent a clustering of the data. In other words, which node will correspond to which cluster (or object class) is dynamically decided during the training process. To achieve this, we formulate an approach that only needs to modify the cost criterion above the softmax layer of any neural network which was designed for a classification task. We therefore present a new pairwise cost function for clustering that can take the place of, or be combined with, the traditional supervised classification loss functions. This flexibility allows the network to use both types of information, depending on which is available.

3.3.1 KLD-based Contrastive Loss (KCL)

While the output of the traditional softmax layer represents the probability that a sample belongs to the class labels (or clusters in our problem), the outputs of the whole softmax layer could be viewed as the distribution of possible clusters given a sample (Figure 3.1a). If the data contain the same semantic class, such as the number of hand-written digits, then the distributions between the softmax output for a similar pair should be similar. Conversely, the distribution over the class labels should be dissimilar if the pair belongs to

different clusters (Figure 3.1b). The similarity between distributions could be evaluated by statistical distance such as Kullback-Leibler divergence (KLD). Traditionally this can be used to measure the distance between the output distribution and ground truth distribution. In our case, however, it can instead be used to measure the distance between the two output distributions given a pair of instances (x_p, x_q).

Their corresponding output distributions are defined as $\mathcal{P} = f(x_p)$ and $\mathcal{Q} = f(x_q)$, while f is the neural network with k outputs. The cost of a similar pair is described as :

$$D_{\mathrm{KL}}(\mathcal{P}^\star||\mathcal{Q}) = \sum_{c=1}^{k} p_c log(\frac{p_c}{q_c}) \tag{3.1}$$

$$\mathcal{L}(x_p, x_q)^+ = D_{\mathrm{KL}}(\mathcal{P}^\star||\mathcal{Q}) + D_{\mathrm{KL}}(\mathcal{Q}^\star||\mathcal{P}) \tag{3.2}$$

The cost $\mathcal{L}(x_p, x_q)^+$ is symmetric w.r.t. x_p, x_q, in which $\mathcal{P}^\star$ and $\mathcal{Q}^\star$ are alternatively assumed to be constant. Each KL-divergence factor $D_{\mathrm{KL}}(\mathcal{P}^\star||\mathcal{Q})$ becomes a unary function whose gradient is simply $\partial D_{\mathrm{KL}}(\mathcal{P}^\star||\mathcal{Q})/\partial\mathcal{Q}$.

If x_p, x_q comes from a pair which is dissimilar, their output distributions are expected to be different, which can be defined as a hinge-loss function:

$$\mathcal{L}(x_p, x_q)^- = L_h(D_{\mathrm{KL}}(\mathcal{P}^\star||\mathcal{Q}), \sigma) + L_h(D_{\mathrm{KL}}(\mathcal{Q}^\star||\mathcal{P}), \sigma) \tag{3.3}$$

$$L_h(e, \sigma) = max(0, \sigma - e) \tag{3.4}$$

The total loss $\mathcal{L}(x_p, x_q)$ combines both losses multiplied by an indicator function I_s, which equals to one when (x_p, x_q) is a similar pair and equals to zero when (x_p, x_q) is dissimilar:

$$\mathcal{L}(x_p, x_q) = I_s\mathcal{L}(x_p, x_q)^+ + (1 - I_s)\mathcal{L}(x_p, x_q)^- \tag{3.5}$$

We called Equation 3.5 the KLD-based Contrastive Loss (KCL), which is differentiable thereby it can be optimized with standard back-propagation algorithm. To calculate the derivative of $\mathcal{L}(x_p, x_q)$, it is worth to note that the $\mathcal{P}$ in the first term of equation 3.2 (and

Q in the second term) is regarded as constant instead of variable. Thus, the derivative is computed as:

$$\frac{\partial}{\partial Q_i}\mathcal{L}(x_p, x_q) = \begin{cases} -\frac{P_i}{Q_i} & \text{if} \quad I_s = 1, \\ \frac{P_i}{Q_i} & \text{elseif} \quad D_{\mathrm{KL}}(\mathcal{P}^\star||\mathcal{Q}) < \sigma, \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

$$\frac{\partial}{\partial P_i}\mathcal{L}(x_p, x_q) = \begin{cases} -\frac{Q_i}{P_i} & \text{if} \quad I_s = 1, \\ \frac{Q_i}{P_i} & \text{elseif} \quad D_{\mathrm{KL}}(\mathcal{Q}^\star||\mathcal{P}) < \sigma, \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

3.3.2 Efficient Implementation to Utilize Pairwise Constraints

Equation 3.5 is in the form of contrastive loss that is suitable to be trained with Siamese networks [80]. However, when the amount of pairwise constraints increases, it is not efficient to enumerate all pairs of data and feed them into Siamese networks. Specifically, if there is a mini-batch that has pairwise constraints between any two samples, the number of pairs that have to be fed into the networks will be $n(n-1)/2$ where n is mini-batch size. However, a redundancy occurs when a sample has more than one constraint associated with it. In such cases the sample will be fed-forward multiple times. However, feeding forward once for each sample is sufficient for calculating the pairwise cost in a mini-batch. Figure 3.2c demonstrates an example for the described situation. The data with index 1 and 3 are fed-forward twice in vanilla Siamese networks to enumerate the three pairwise relationships: (1,2),(1,3), and (3,4). To avoid the redundancy of computation, we apply a strategy of enumerating the pairwise relationships only in the cost layer, instead of instantiating the Siamese architecture. This strategy simplified the implementation of neural networks which utilize pairwise relationship. Our proposed architecture is shown in the Figure 3.2b. The pairwise constraints only need to be presented to the cost layer in the format of tuples $T : (i, j, relationship)$ where i and j are the index of sample inside the mini-batch

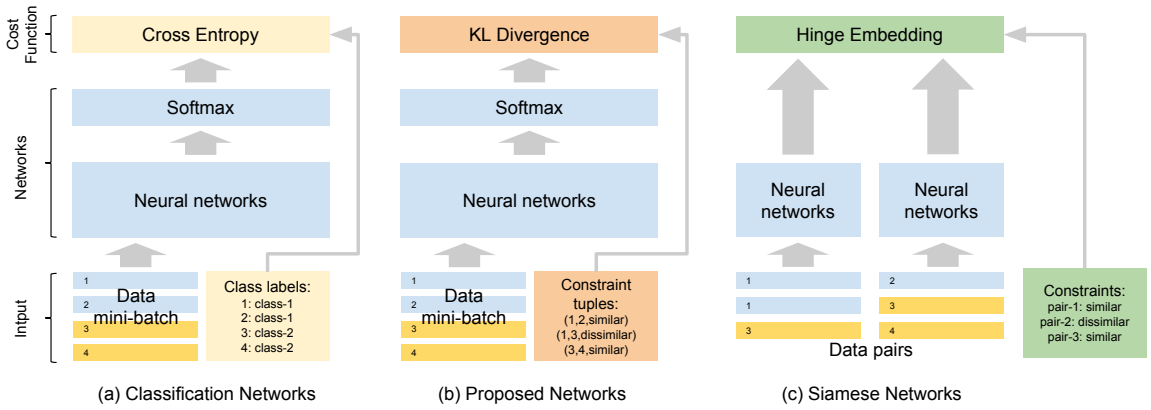


Figure 3.2: The comparison between (a) classification networks, (b) our proposed networks, and (c) Siamese networks. The parts that differ across architectures are shown with distinct colors. In (a) and (b), the numbers in the data represent the index of the input data in a mini-batch.

and $relationship$ indicates similar/dissimilar pair. Each input data is therefore only fed-forward once in a mini-batch and its full/partial pairwise relationships are enumerated as tuples.

Concretely, the gradients for back-propagation in a mini-batch are calculated as:

$$\frac{\partial}{\partial f(x_i)}\mathcal{L} = \sum_{\forall j;(i,j)\in T} \frac{\partial}{\partial f(x_i)}\mathcal{L}(f(x_i), f(x_j)). \tag{3.8}$$

One could see our proposed architecture (Figure 3.2b) is highly similar to the standard classification networks (Figure 3.2a). As a result of this design, ideas in the above two sections could be easily implemented as a learning criterion module in popular deep learning frameworks such as PyTorch [93].. Then a network could be switched to either classification mode or clustering mode by simply changing the cost criterion. We call our network implementation strategy together with KCL as the *NNclustering* method.

3.4 Experiments

We evaluate the proposed clustering objective on the MNIST [94] and CIFAR-10 [95] datasets. The two datasets are both normalized to zero mean and unit variance. The con-

volutional neural networks architecture used in these experiments is similar to LeNet [94]. The network has 20 and 50 5x5 filters for its two convolution layers with batch normalization [96] and 2x2 max-pooling layers. We use the same number of filters for both MNIST and CIFAR-10 experiments. The two subsequent fully connected layers have 500 and 10 nodes. Both convolutional and the first fully connected layers are followed by rectified linear units. The only hyper-parameter in our cost function is the $margin$ in equation 3.4. The margin was chosen by cross-validation on the training set. To minimize the clustering objective, we applied mini-batch stochastic gradient descent.

3.4.1 Clustering with Partial Constraints

The experiments in this section seek to demonstrate how the approach works with partial constraints. In this case, we use a clustering metric to demonstrate how good the resulting clustering is. The constraints are uniformly sampled from the full set, i.e, the number of full constraints is $n(n-1)/2$, where n is the size of training set. The pairwise relationship is converted from the class label. If a pair has the same class label, then it is a similar pair, otherwise it is dissimilar. We did not address the fact that the amount of dissimilar pairs usually dominates the pairwise relationship (which is more realistic in many application domains), especially when the number of classes is large. In our experiments for this section, the ratio between the number of similar and dissimilar pairs is roughly 1:9.

We evaluate the resulting clusters with the purity measure and normalized mutual information (NMI) [97]. The index of cluster for each sample is obtained by feed-forwarding the training/testing data into the trained networks. Note that we collect the clustering results of training data after the training error has converged, i.e, feed the training data one more time to collect the outputs after the training phase. We picked the networks which have the lowest training loss among five random restarts while the set of constraints being kept the same.

Figure 3.3 shows that on MNIST the clustering could still achieve high accuracy when

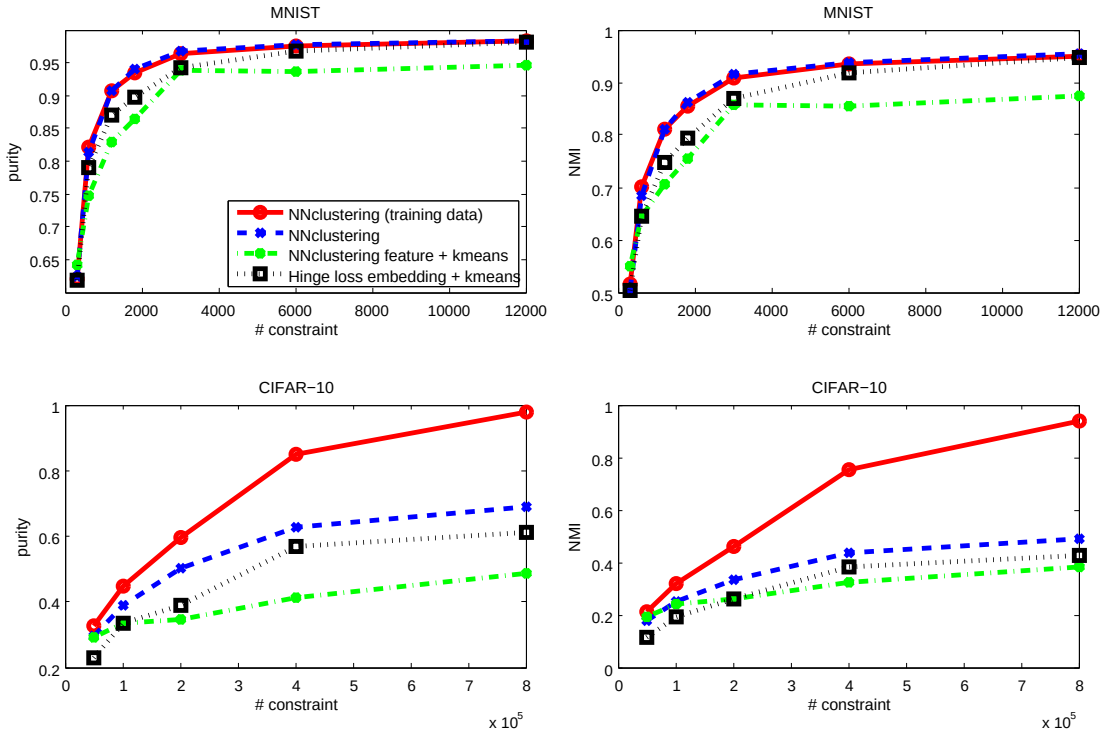


Figure 3.3: The results of clustering with partial pairwise constraints. The axis with #constraint is the number of sampled pairwise relationship in the training data. The clustering and training is simultaneously applied on the training data (red line). The testing data (for blue, green, black lines) is used to validate if the feature space learned during clustering has generalizability. *NNclustering* is the neural network optimized with KCL. The *NNclustering feature + kmeans* uses the outputs at the last hidden layer (500-D) as the input for k-means. The baseline networks (black line) were trained with hinge loss of Euclidean distance. The evaluation metric in the first column is purity, while the second column shows the NMI score. Note that the first row uses MNIST dataset, while the second row uses CIFAR-10.

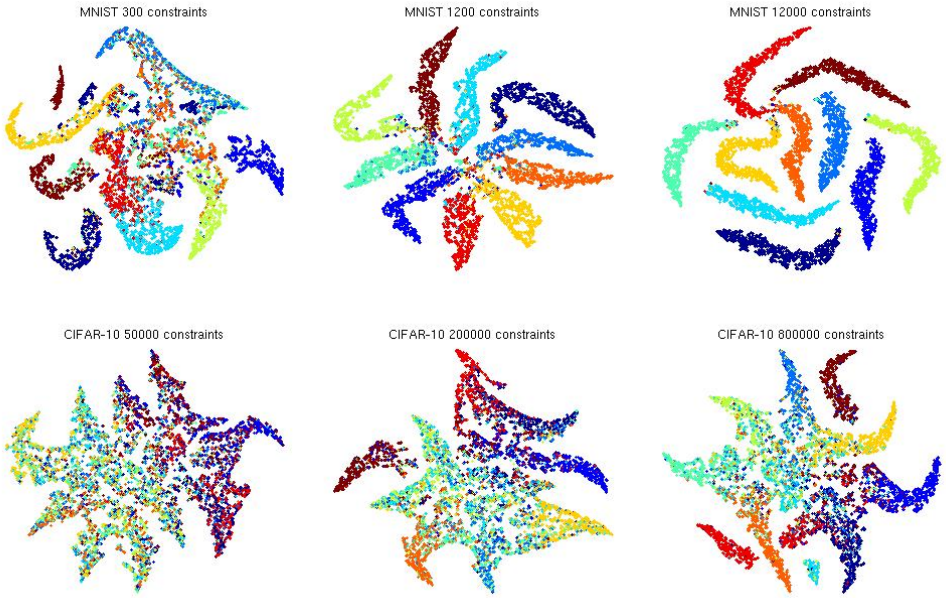


Figure 3.4: The visualization of clustering. The figure was created by using the outputs of the softmax layer as the input for t-SNE [98]. Only testing data are shown. The networks used in the first row are trained with 300, 1200, and 12000 pairs of constraints in MNIST training set. The second row is trained with 50k, 200k, and 800k constraints in CIFAR-10.

constraints are extremely sparse. With merely 1200 constraints, which were randomly sampled from the pairwise relationship of the full (60000 samples) training set, it achieves $>$0.9 purity and $>$0.8 NMI scores. Note that a training sample without any constraints associated with it has no contribution to the training. Thus, the scheme is not the same as the semi-supervised clustering framework in previous works [18, 90, 91] where their unlabeled data contribute to calculating the centers of the clusters. The lack of explicit cluster centers provides the flexibility to learn more complex non-linear representations, so the proposed algorithm could still predict the cluster of unseen data without knowing the cluster centers.

To demonstrate the advantage of performing joint clustering and feature learning, we also applied the k-means algorithm with the features learned at the last hidden layer, which has 500 dimensions. The k-means algorithm used Euclidean or cosine distance and was deployed with 50 random restarts on the testing set. We report the clustering results of

k=10 which has the lowest sum of point-to-centroid distances among 50 restarts. Since the dimensionality is relatively high, the performance of using Euclidean and cosine distance showed minor difference. The results in Figure 3.3 show that the jointly trained last layer utilize the outputs of last hidden layer much better than k-means.

To construct the baseline approach, we use the common strategy of training a Siamese networks with standard hinge loss embedding criteria in *torch nn* package, then perform k-means on the networks' outputs. The baseline networks have the same architecture except the softmax layer and the loss function. Figure 3.3 shows that the proposed clustering framework beats the baseline with a significant margin when the number of constraints is few in the easy dataset (purity is $\sim 5\%$ better in MNIST) or when the dataset is harder (purity is $15 \sim 50\%$ better in CIFAR-10).

The experiments with CIFAR-10 provides some idea of how the approach works on a more difficult dataset. The required constraints to achieve reasonable clustering is much higher. Eight constraints/sample (400,000 total constraints) is required to reach a 0.8 purity score with the same network. The performance on unseen data is also degraded because the networks is over-fitting to the constraints. The degradation could possibly be mitigated by adding more regularization terms such as dropout. While any general regularization strategy could be applied in the proposed scheme, we do not address this in this work. Nevertheless, the clustering on the training set is still effective with sparse constraints, e.g., it is able to reach a purity of ~ 1 with only 16 constraints/sample on CIFAR-10. The visualization in Figure 3.4 provides more intuition about the clustering results trained with different numbers of constraints.

3.4.2 Robustness of Clustering

3.4.2.1 Adding Noise

Noisy constraints are likely to occur when the pairwise relationships are generated in an automatic/unsupervised way. We simulated this scenario by flipping the sampled pairwise

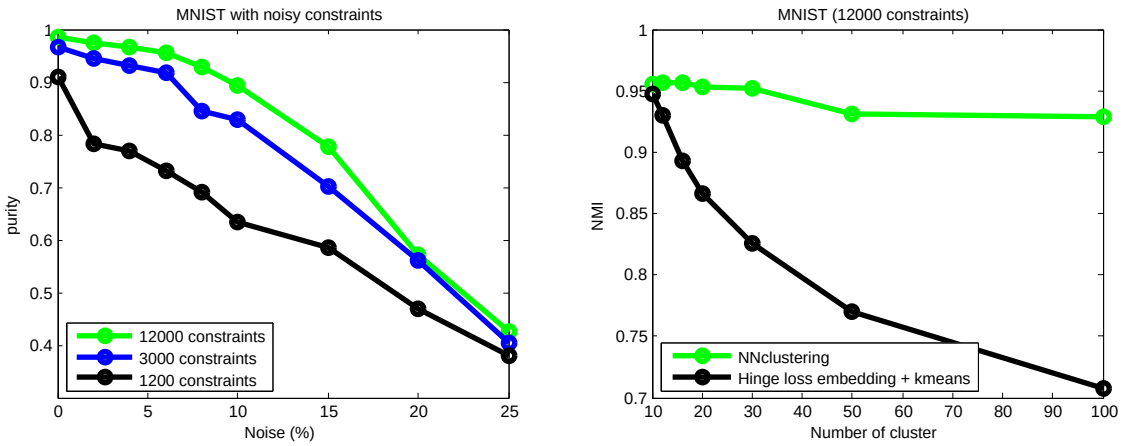


Figure 3.5: The robustness evaluation of the proposed clustering method. Left figure is the result of adding noisy constraints into MNIST, while the right figure simulates the case when the number of clusters is unknown.

relationship. Since the ratio of similar pair and dissimilar pair is 1:9, adding 10% noise will introduce equal amount of false-similar pair as the amount of true-similar pair. The clustering performance in Figure 3.5 (left) shows the reasonable tolerance against noise. We would like to point out that when noise is less than 10%, the performance degradation is reduced when the number of constraints increased. This means that the proposed method could achieve higher performance by adding more pairwise information while keeping the ratio of noise the same. Real applications would benefit from this property since adding more weakly labeled data is cheap and the noise level of automatically generated constraints are usually the same.

3.4.2.2 Changing Number of Clusters

Another common scenario is that the number of target clusters is unknown. Since the purity metric is not sensitive to the number of clusters, NMI is more appropriate in this evaluation. We performed the experiment with 12,000 constraints for training, which include $\approx$1200 similar pairs in MNIST. The testing results in the right of figure 3.5 show that the proposed method is almost not affected by increasing the number of output cluster nodes that the network can use. Even in the condition of 100 clusters (by setting 100 output nodes in our

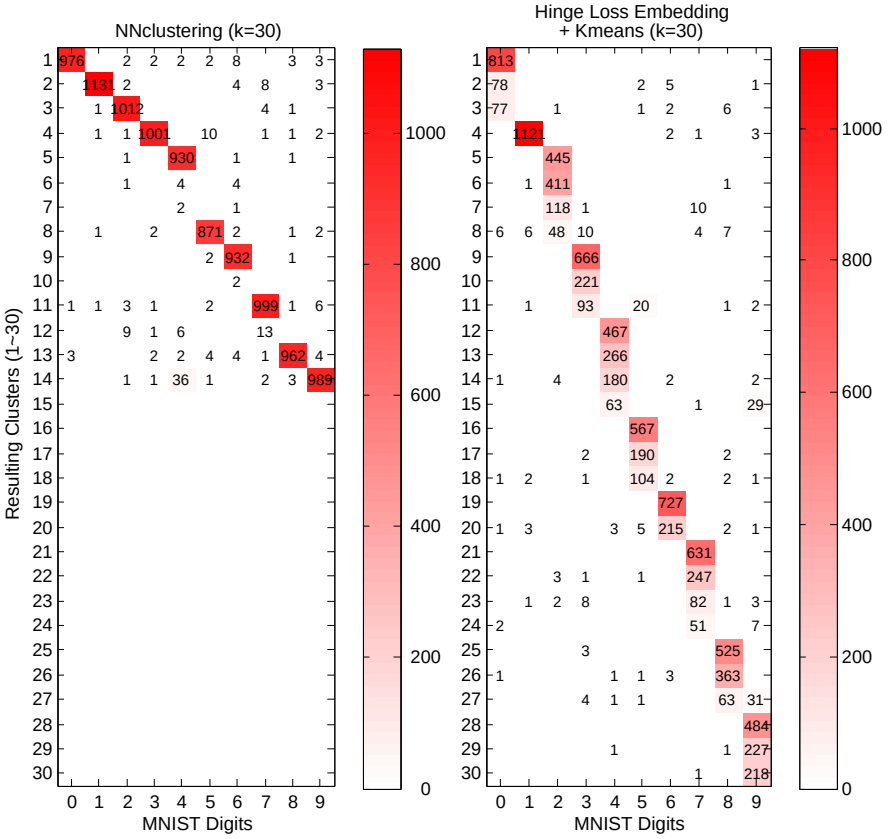


Figure 3.6: The contingency tables of resulting clusters. It only shows k=30 (same experiment in the right part of figure 3.5) for the ease of visualization. NNclustering produces similar result even when k=100. The numbers in the table show the amount of samples assigned to the cluster, while the blank rows indicate empty clusters. Higher numbers in fewer positions is the preferred result for clustering.

networks), the performance only decreases by a very small amount. In fact, in figure 3.6 most of the data were assigned to $\approx$10 major clusters and left other clusters being empty. In contrast, the kmeans-based approach (hinge loss embedding + kmeans) is susceptible to the number of clusters and usually divide a class into many small clusters.

3.4.2.3 Combination of all factors

This section investigates when three factors (noise, number of constraints, and number of clusters) are combined, how does the KCL perform.

Experiment setup: The networks were randomly initialized and the clustering training was run five times under each combination of factors; we show the best final results, as is usual in the random restart regime. The mini-batch size was set to 256, thus up to 65536 pairs were presented to the KCL per mini-batch if using full density (D=1). There were 235 mini-batches in an epoch and the optimization proceeded for 15 epochs. The clustering loss was minimized by stochastic gradient descent with learning rate 0.1 and momentum 0.9. The predicted cluster was assigned at the end by forwarding samples through the clustering networks. The best result in the five runs was reported.

To simulate the noisy similarity prediction, the label of pairs were flipped according to the designated recall. For example, to simulate a 90% recall of similar pair, 10% of the ground truth similar pair in a mini-batch were flipped. The precision of similar/dissimilar pairs is a function of the recall of both type of pairs, thus controlling the recall is sufficient for the evaluation. The recalls for both similar and dissimilar pairs were gradually reduced from one to zero at intervals of 0.1.

Results: The resulting performance w.r.t different values of recall, density, and the number of clusters is visualized in Figure 3.5. A bright color means a high NMI score and is desired. The larger the bright region, the more robust the clustering is against the noise of similarity prediction. The ACC score shows almost the same trend and is thus not shown here.

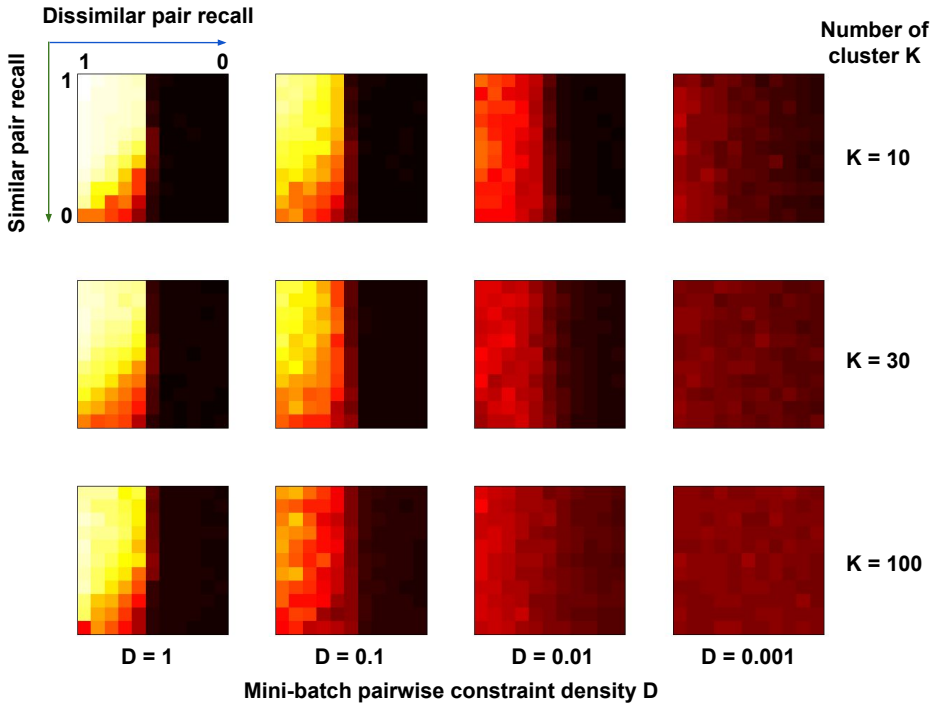


Figure 3.7: The clustering performance with different pairwise density and number of clusters. A bright color means that the NMI score is close to 1 while black corresponds to 0. The density is defined as a ratio compared to the total number of pair-wise combinations in a mini-batch. The number of clusters defines the final softmax output dimensionality. In each sub-figure, we show how the scores change w.r.t. the similar pair recall and dissimiliar pair recall.

How does similarity prediction affect clustering? Looking at the top-left heat map in figure 3.7, which has $D = 1$ and 10 clusters, it can be observed that the NMI score is very robust to low similar pair recall, even lower than 0.5. For recall of dissimilar pairs, the effect of recall is divided at the 0.5 value: the clustering performance can be very robust to noise in dissimilar pairs if the recall is greater than 0.5; however, it can completely fail if the recall is below 0.5. For similar pairs, the clustering works on a wide range of recalls when the recall of dissimilar pairs is high.

In practical terms, robustness to the recall of similar pairs is desirable because it is much easier to predict dissimilar pairs than similar pairs in real scenarios. In a dataset with 10 categories e.g. CIFAR-10, we can easily get 90% recall for dissimilar pairs with a purely random guess if the number of classes is known, while the recall for similar pairs will be 10%.

How does the density of the constraints affect clustering? We argue that the density of pairwise relationships is the key factor in improving the robustness of clustering. The density $D = 1$ means that the clustering loss utilizes every pair in a mini-batch. For density $D = 0.1$, it means only 1 out of 10 possible constraints is used. We could regard the higher density as better utilization of the pairwise information in a mini-batch; thus, more learning instances contribute to the gradients at once. Consider a scenario where there is one sample associated with 5 true similar pairs and 3 false similar pairs. In such a case, the gradients introduced by the false similar pairs have a higher chance to be overridden by true similar pairs within the mini-batch; thus, the loss can converge faster and is less affected by errors. In Figure 3.7, we could see when density decreases, the size of the bright region shrinks significantly.

In our implementation, enumerating the full pairwise relationships introduces negligible overhead in computation time using GPU. Although there is overhead for memory consumption, it is limited because only the vector of predicted distributions has to be enumerated for calculating the clustering loss.

The effect of varying the number of Clusters: In the MNIST experiments, the number of categories is 10. We augment the softmax output number up to 100. The rows of figure 3.7 show that even when the number of output categories is significant larger than the number of true object categories, e.g. $100 > 10$, the clustering performance NMI score only degrades slightly.

3.4.3 Clustering versus Classification

The final set of experiments compares the accuracy of our approach with a pure classification task in order to get an upper bound of performance (since full labels can be used to create a full set of constraints) and see whether our approach can leverage pairwise constraints to achieve similar results. To make the results of clustering (contrastive loss) comparable to classification (cross-entropy loss), the label of each cluster is obtained from the training set. Specifically, we make the number of output nodes to be the same as the true number of classes, thus we could assign each output node with a distinct label using the optimal assignment, Hungarian algorithm [99]. The results in Table 3.1 show our cost function achieved slightly higher or comparable accuracy in most of the experiment settings. The exception is MNIST with 6 samples/class. The reason is that the proposed cost function creates more local minimum. If the training data is too few, then the training will be more likely to be trapped in certain local minimum. Note that we also applied a random restart strategy (randomly initializing the parameters of the network) to find a better clustering result based on the training set, which is a common strategy used in typical clustering procedures. We ran 5 randomly initialized networks to perform clustering and chose the network that had the highest training accuracy and then used the resulting network to predict the clusters on the testing set.

We also performed the experiments using the same architecture applied to a harder dataset, i.e, CIFAR-10. We did not pursue optimal performance on the dataset, but instead used it to compare the performance difference of learning between the classification and

Table 3.1: Comparing the testing accuracy between classification and clustering using same networks architecture. The clustering is trained with full pairwise relationships obtained from ground-truth class labels. The separated testing set (10,000 samples) is used in this evaluation.

Training approach	Classification	Clustering
Training data:		
MNIST 6 sample/class	**82.4%**	79.4%
MNIST 60 sample/class	94.7%	**95.1%**
MNIST 600 sample/class	98.3%	**98.8%**
MNIST full (≈6000 sample/class)	99.4%	**99.6%**
Training data:		
CIFAR-10 5 sample/class	21.3%	**22.0%**
CIFAR-10 50 sample/class	34.6%	**37.0%**
CIFAR-10 500 sample/class	**55.0%**	53.2%
CIFAR-10 full (5000 sample/class)	**73.7%**	73.4%

clustering networks. The results show that they are fully comparable. Since CIFAR-10 is a much more difficult dataset compared to MNIST, the overall drop of accuracy on CIFAR-10 is reasonable. Even in the extreme case when the number of training samples is small, the proposed architecture and cost function proved effective.

3.4.4 Limitation

We found that KCL may have a hard time to converge when the number of parameters in the model is large, or when there is a large number of classes or clusters. It may because the KCL formulation introduces plateaus or local minimums to the optimization. We have qualitative and quantitative analysis in Chapter 5 to demonstrate this property, and compare KCL with our other methods. This limitation can be mitigated by having better weight initialization (*e.g.* using pre-trained weights) or by training for a longer time.

3.5 Conclusion

We introduce a novel framework and construct a cost function for training neural networks to both learn the underlying features while, at the same time, clustering the data in the resulting feature space. We show strong results compared to traditional K-means clustering, even when it is applied to a feature space learned by a Siamese network. Our robustness analysis not only shows good tolerance to noise, but also demonstrates the significant advantage of our method when the number of clusters is unknown. We also demonstrate that, using only pairwise constraints, we can achieve equal or slightly better results than when explicit labels are available and a classification criterion is used. In addition, our approach is both easy to implement for existing classification networks (since the modifications are in the cost layer) and can be efficiently implemented. The work of this section is published in [9].

CHAPTER 4

TRANSFER PAIRWISE SIMILARITY ACROSS TASKS AND DOMAINS

4.1 Introduction

In this chapter, we propose to use predictive pairwise similarity as the knowledge that is transferred and formulate a learnable objective function to utilize the pairwise information in a fashion similar to constrained clustering (Chapter 3). We then provide the methodologies to deploy the objective function in both cross-task and cross-domain scenarios with deep neural networks. The work of this chapter is published in [11], elaborated in below.

Supervised learning has made significant strides in the past decade, with substantial advancements arising from the use of deep neural networks. However, a large part of this success has come from the existence of extensive labeled datasets. In many situations, it is not practical to obtain such data due to the amount of effort required or when the task or data distributions change dynamically. To deal with these situations, the fields of transfer learning and domain adaptation have explored how to transfer learned knowledge across tasks or domains. Cross-domain transfer learning focuses on cases where the distributions of the features and labels have changed, but the task is the same (e.g., classification across datasets with the same categories). Cross-task transfer learning strategies, on the other hand, have been widely adopted especially in the computer vision community where features learned by a deep neural network on a large classification task have been applied to a wide variety of other tasks [29].

Most of the prior cross-task transfer learning works, however, require labeled target data to learn classifiers for the new task. If labels of the target data are absent, there is little choice other than to apply unsupervised approaches such as clustering on the target data with pre-trained feature representations. In this chapter, we focus on the question of what

can be transferred (besides features) to support both cross-domain and cross-task transfer learning. We address it with a learned similarity function as the fundamental component of clustering. Clustering can then be realized using a neural network trained using the output of the similarity function, which can be successfully used to achieve both cross-task and cross-domain transfer.

The key idea is to formulate the clustering objective to use a learnable (and transferable) term, which in our proposed work is a similarity prediction function. Our proposed objective function can be easily combined with deep neural networks and optimized end-to-end. The features and clustering are optimized jointly, hence taking advantage of such side information in a robust way (investigated in Chapter 3). Using this method, we show that unsupervised learning can benefit from learning performed on a distinct task, and demonstrate the flexibility of further combining it with a classification loss and domain discrepancy loss.

The experimental results for cross-task learning on Omniglot and ImageNet show that we can achieve state of the art clustering results with predicted similarities. On the standard domain adaptation benchmark Office-31 dataset, we demonstrate improvements over state-of-art even when not performing any explicit domain adaptation, and further improvements if we do. Finally, on another domain adaptation task, SVHN-to-MNIST, our approach using Omniglot as the auxiliary dataset achieves top performance with a large margin.

4.2 The Transfer Learning Tasks

To define the transfer learning problem addressed in this chaper, we follow the notations used by [28]. The goal is to transfer knowledge from source data $S = (X_S, Y_S)$, where X_S is the set of data instances and Y_S is the corresponding categorical labels, to a target data noted as $T = (X_T, Y_T)$. The learning is unsupervised since Y_T is unknown. The scenario is divided into two cases. One is $\{Y_T\} \neq \{Y_S\}$, which means the set of categories are not the same and hence the transfer is across tasks. The second case is $\{Y_T\} = \{Y_S\}$, but with a domain shift. In other words, the marginal probability distributions of the input data

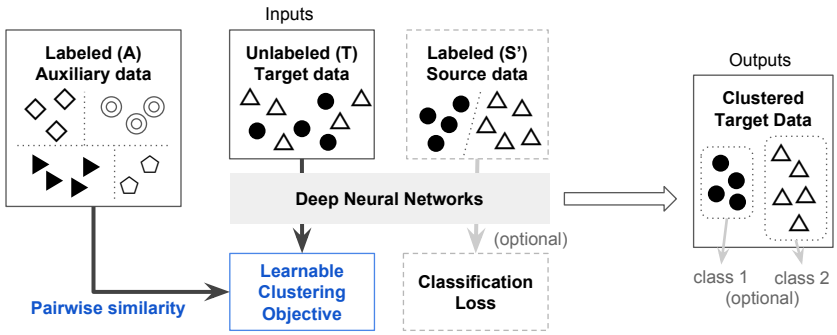


Figure 4.1: Overview of the transfer scheme with a learnable clustering objective (LCO). The LCO and pairwise similarity are the two key components of our approach and are described in section 4.4. The dashed rectangles and light gray arrows are only available in cross-domain transfer. Details are described in section 4.2.

are different, i.e., $P(X_T) \neq P(X_S)$. The latter is a cross-domain learning problem also called transductive learning. Domain adaptation approaches, which have gained significant attention recently, belong to the second scenario.

To align with common benchmarks for evaluating transfer learning performance, we further use the notion of an auxiliary dataset and split the source data into $S = S' \cup A$. $S' = (X'_S, Y'_S)$ which is only present in the cross-domain transfer scheme and has $\{Y'_S\} = \{Y_T\}$. $A = (X_A, Y_A)$ is the auxiliary dataset which has a large amount of labeled data and potentially categories as well, and may or may not contain the categories of Y_T. For the cross task scenario, only A and unlabeled T are included, while cross-domain transfer involves A, S', and T. In the following sections we use the above notations and describe the two transfer learning tasks in detail. Figure 4.1 illustrates how our approach relates to both tasks.

Transfer across tasks: If the target task has different categories, we cannot directly transfer the classifier from source to target, and there is no labeled target data to use for fine-tuning transferred features. Here we propose to first reduce the categorization problem to a surrogate same-task problem. We can directly apply transductive transfer learning [28] to the transformed task. The cluster structure in the target data is then reconstructed using the predictions in the transformed task. See figure 4.2 for an illustration.

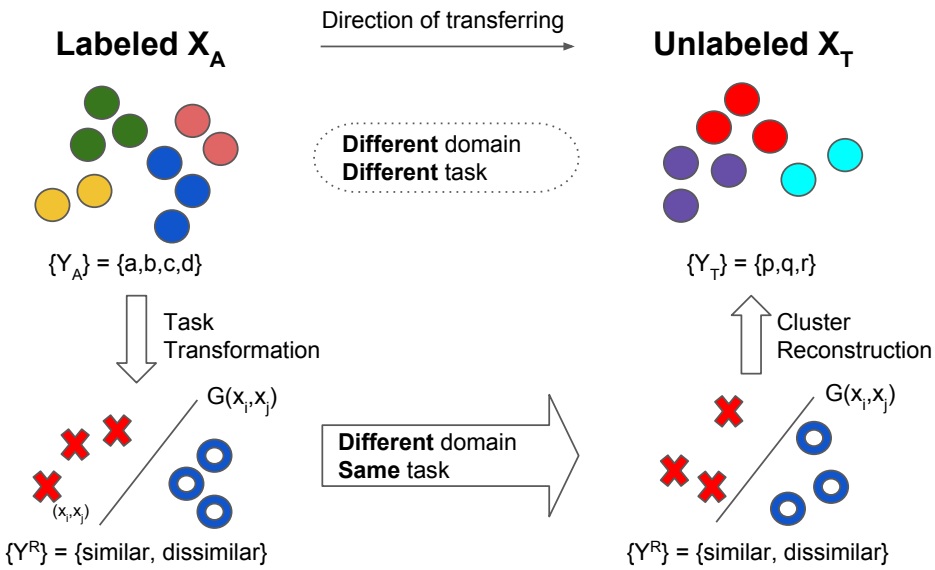


Figure 4.2: The concept for reconstructing clusters of unseen categories. The proposed approach follows the arrows in the counter-clockwise direction, which converts cross-task transfer learning to cross-domain transfer learning. The colors of dots represent data of different categories. The hollow circle and cross symbol represent similar and dissimilar data pairs. The G function and the cluster reconstruction (via constrained clustering) are the two key components in the diagram.

The source involves a labeled auxiliary dataset A and an unlabeled target dataset T. Y_T is the target that must be inferred. In this scenario the set of categories $\{Y_A\} \neq \{Y_T\}$, and Y_T is unknown. We first transform the problem of categorization into a pairwise similarity prediction problem. In other words, we specify a transformation function R such that $R(A) = (X_A^R, Y^R)$, and $X_A^R = \{(x_{A,i}, x_{A,j})\}_{\forall i,j}$ contains all pairs of data, where $\{Y^R\} = \{dissimilar, similar\}$. The transformation on the labeled auxiliary data is straightforward. It will be $similar$ if two data instances are from the same category, and vice versa. We then use it to learn a pairwise similarity prediction function $G(x_i, x_j) = y_{i,j}$. By applying G on T, we can obtain $G(x_{T,i}, x_{T,j}) = y_{T,i,j}$. The last step is to infer Y_T from $Y_T^R = \{y_{T,i,j}\}_{\forall i,j}$, which can be solved using constrained clustering algorithms. Note that since the actual Y_T is unknown, the algorithm only infers the indices of categories, which could be in arbitrary order. The resulting clusters are expected to contain coherent semantic categories.

Transfer across domains: The problem setting we consider here is the same as unsupervised domain adaptation. Following the standard evaluation procedure [43, 38], the labeled datasets A is ImageNet and S' is one domain in the Office-31 dataset. The unlabeled T is another domain in Office-31. The goal is to enhance classification performance on T by utilizing A, S', and T together.

4.3 Related Work

The major part of the related work for this chapter has been discussed in Chapter 2. Here we emphasize how this chapter is different from previous works regarding the two learning scenarios.

Unsupervised Cross-task transfer learning: Features learned when trained for ImageNet classification [100] have boosted the performance of a variety of vision tasks in a supervised setting. For example, new classification tasks [29, 30], object detection [31], semantic segmentation [32], and image captioning [33]. Translated Learning [101] has an unsupervised setting similar to ours, but it focuses only on transferring features across tasks. Our work explores how learning could benefit from transferring pairwise similarity in an unsupervised setting. Note that the works [102, 103] which followed on our work call this setting a *few-shot clustering* [104], since the number of data in the target dataset can be small in our experiment.

Unsupervised Cross-domain transfer learning: Cross-domain Transfer Learning also known as domain adaptation [28], there has recently been a large body of work dealing with domain shift between image datasets by minimizing domain discrepancy [35, 37, 38, 39, 40, 41, 42, 43, 44, 45]. We address the problem in a complementary way that transfers extra information from the auxiliary dataset and show a larger performance boost with further gains using an additional domain discrepancy loss.

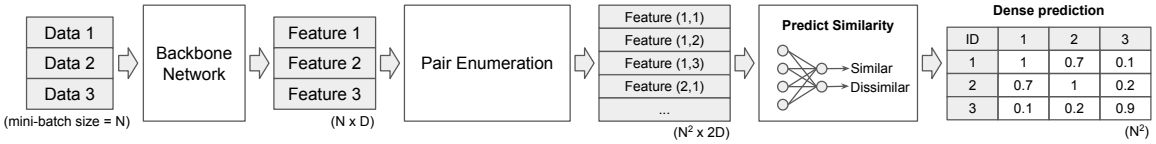


Figure 4.3: The similarity prediction network (SPN, or called the G function).

4.4 The Learnable Clustering Objective (LCO)

The key to our approach is the design of a learning objective that can use (noisy) predicted pairwise similarities, and is inspired from constrained clustering which involves using pairwise information in the loss function. Although many constrained clustering algorithms have been developed, few of them are scalable with respect to the number of pairwise relationships. Further, none of them can be easily integrated into deep neural networks. The KCL (equation 3.5) in Chapter 3 demonstrates strong robustness against noisy similarity thereby we extend it usage to the transfer learning in this chapter. Here we replace the indicator function I_s in KCL by G, the similarity prediction function $G(x_i, x_i) \in \{0, 1\}$, which is introduced in section 4.2, :

$$\mathcal{L}(x_i, x_i) = G(x_i, x_i)\mathcal{L}(x_i, x_i)^+ + (1 - G(x_i, x_i))\mathcal{L}(x_i, x_i)^- \tag{4.1}$$

We refer to the form of equation 4.1 as LCO. Function G (Figure 4.3) is the learnable part that utilizes prior knowledge and is trained with auxiliary dataset A before optimizing LCO. Two particular characteristics of the clustering criterion are worth mentioning: (1) there is no need to define cluster centers, and (2) there is no predefined metric applied on the feature representation. Instead, the divergence is calculated directly on the cluster assignment; therefore, both feature representation and clustering are jointly optimized using back-propagation through the deep neural networks.

4.4.1 The Pairwise Similarity Prediction Network

Although there is no restriction of how G should be constructed, we choose deep convolution neural networks due to their efficiency in vision tasks. We design a network architecture inspired from [105]. While they use it to predict image patch similarity, we use it to predict image-level semantic similarity. However, the Siamese architecture used in [105] is not efficient in both training and inference, especially when pairwise information is dense. Therefore, instead of using Siamese architecture, we keep the single column backbone but add a *pair-enumeration layer* on top of the feature extraction network, which is similar to the efficient implementation of KCL described in Section 3.3.2. The pair-enumeration layer enumerates all pairs of feature vectors within a mini-batch and concatenates the features. Suppose the input of the layer is 10×512 with the mini-batch size 10 and the feature dimension 512; then the output of the pair-enumeration layer will be 100×1024 (self-pairs included).

The architecture is illustrated in figure 4.3. We add one hidden fully connected layer on top of the enumeration layer and a binary classifier at the end. We use the standard cross-entropy loss and train it end-to-end. The supervision for training was obtained by converting the ground-truth category labels into binary similarity, i.e., if two samples are from the same class then their label will be *similar*, otherwise *dissimilar*. The inference is also end-to-end, and it outputs predictions among all similarity pairs in a mini-batch. The output probability $g \in [0, 1]$ with 1 means more similar. We binarize g at 0.5 to obtain discrete similarity predictions. In the following sections, we simplified the notation of the pairwise similarity prediction network as G. Once G is learned, it then works as a static function in our experiments.

4.4.2 The Objective Function with Dense Similarity Prediction

Since the pairwise prediction of a mini-batch can be densely obtained from G , to efficiently utilize the pair-wise information without forwarding each data multiple times we also add a

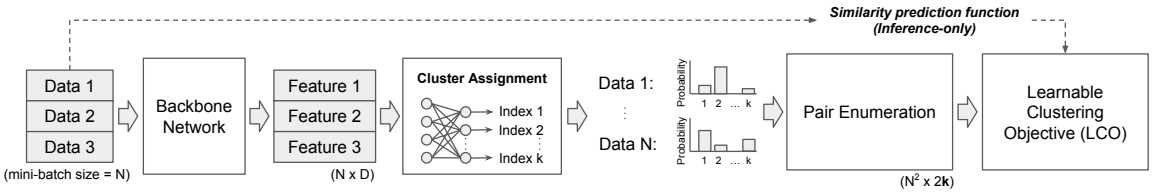


Figure 4.4: The constrained clustering network (CCN) for transfer learning across tasks. The input is unlabeled target data T. The cluster assignment block contains two fully connected layers and has the number of output nodes equal to k. The f described in section 4.4 is the backbone network plus the cluster assignment block. To optimize LCO, the full pipeline in the diagram is used. After the optimization, it uses another forward propagation with only f to obtain the final cluster assignment.

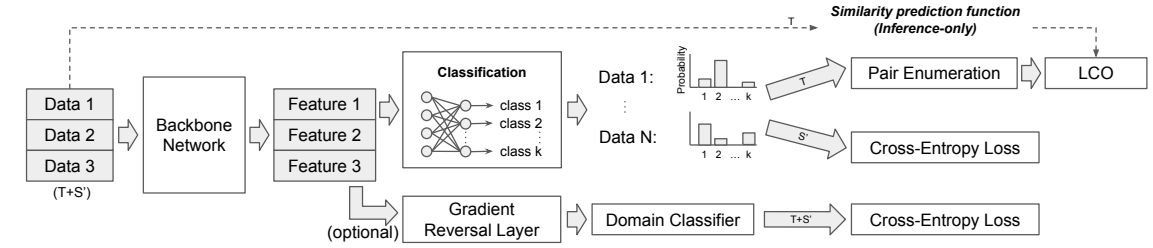


Figure 4.5: The network for transfer learning across domains. The input is the mix of S' and T. The architecture is a direct extension of CCN. We use CCN$^+$ to represent the mandatory parts (upper branch) which implements equation 4.3. CCN^{++} includes the domain adaptation method (optional branch).

pair-enumeration layer illustrated in Figure 4.3 with equation 4.1. In this case, the outputs of softmax are enumerated in pairs. Let D be the set of all tuples (p, q), while p and q are the indices of a sample in a mini-batch. The dense clustering loss $\mathcal{L}_d$ for a mini-batch is calculated by:

$$\mathcal{L}_d = \sum_{\forall (i,j) \in D} \mathcal{L}(x_i, x_i). \tag{4.2}$$

We use $\mathcal{L}_d$ standalone with deep neural networks to reconstruct semantic clusters and for transfer learning across tasks (figure 4.4). We call the architecture the constrained clustering network (CCN).

4.4.3 Combining with Other Objectives

In the cross-domain case, we additionally have labeled source data. This enables us to use LCO for T while using classification loss for S'. The overall training procedure is similar to previous domain adaptation approaches in that both source and target data are mixed in a mini-batch, but different losses are applied. We denote the source domain images in a mini-batch b as X_{b^S} and the target domain images X_{b^T} with its set of dense pairs D_{b^T}. The loss function $\mathcal{L}_{cd}$ for cross-domain transfer in a mini-batch can then be formulated as:

$$\mathcal{L}_{cd} = \mathcal{L}_{cls} + \mathcal{L}_{cluster} \tag{4.3}$$

$$\mathcal{L}_{cls} = \frac{1}{|b^S|} \sum_{\forall x_i \in X_{b^S}} CrossEntropyLoss(f(x_i)) \tag{4.4}$$

$$\mathcal{L}_{cluster} = \frac{1}{|b^T|^2} \sum_{\forall (i,j) \in D_{b^T}} \mathcal{L}(x_i, x_j) \tag{4.5}$$

The $\mathcal{L}_{cluster}$ and $\mathcal{L}_{cls}$ share the same outputs from network f. Although $\mathcal{L}_{cluster}$ does not force the mapping between clusters and categories, $\mathcal{L}_{cls}$ does. Therefore the learned classifier can be applied on target data directly and picks the maximum probability for the predicted class. Note that in our loss function, there is no term to explicitly match the feature distribution between source and target; it merely transfers more knowledge in the form of constraints to regularize the learning of the classifier. There is also no requirement for the architecture to utilize hidden layers. Therefore our approach has the large flexibility to be combined with other domain adaptation strategies. Figure 4.5 illustrates the architectures CCN^{+} and CCN^{++} used in our cross-domain experiments.

4.5 Experiments

This section contains evaluations with four image datasets and covers both cross-task and cross-domain schemes. The details are described below, and the differences between experimental settings are summarized in Tables 4.1 and 4.2.

Table 4.1: The list of datasets involved in transfer learning experiments. G learns the similarity function from dataset A. The CCN* is optimized with dataset T or T∪S', while CCN* means CCN for cross-task transfer and CCN$^{+/++}$ for cross-domain transfer. The rows for network initialization indicate whether the network has weights initialized by training a classification task with the specified dataset. The weights are randomly initialized if not specified.

Scheme		Cross-Task Transfer		Cross-Domain Transfer	
Experiment Section		Omniglot 4.5.1.1	ImageNet 4.5.1.3	Office-31 4.5.2.1	SVHN-MNIST 4.5.2.3
Datasets	A	$\text{Omniglot}_{\text{bg}}$	ImageNet_{882}	ImageNet_{1000}	$\text{Omniglot}_{\text{bg}}$
	S'	-	-	$\text{Office-31}_{\{a,d,w\}}$	SVHN
	T	$\text{Omniglot}_{\text{eval}}$	ImageNet_{118}	$\text{Office-31}_{\{a,d,w\}}$	MNIST
Network Initialization	G	-	ImageNet_{882}	ImageNet_{1000}	-
	CCN*	-	ImageNet_{882}	ImageNet_{1000}	-

Table 4.2: The list of loss functions used for training networks. The similarity prediction function (network) G uses the cross-entropy (CE) loss with two classes (similar/dissimilar). The training of constrained clustering network (CCN*) involves the combinations of the learnable clustering objective (LCO), cross-entropy, and domain adaptation loss (DA).

	LCO	CE	DA
G		✓	
CCN	✓		
CCN^{+}	✓	✓	
CCN^{++}	✓	✓	✓

4.5.1 Unsupervised Cross-Task Transfer Learning

4.5.1.1 Setup

The **Omniglot** dataset [106] contains 1623 different handwritten characters and each of them has 20 images drawn by different people. The characters are from 50 different alphabets and were separated to 30 background sets $Omniglot_{bg}$ and 20 evaluation sets $Omniglot_{eval}$ by the author. We use the $Omniglot_{bg}$ as the auxiliary dataset (A) and the $Omniglot_{eval}$ as the target data (T). The total number of characters in $Omniglot_{bg}$ is 964, which can be regarded as the number of categories available to learn the semantic similarity. The goal is to cluster $Omniglot_{eval}$ to reconstruct its semantic categories, without ever having any labels.

The G function has a backbone neural network with four 3x3 convolution layers followed by 2x2 max-pooling with stride 2. All hidden layers are followed by batch normalization [96] and rectified linear unit. To prepare the inputs for training, the images from $Omniglot_{bg}$ were resized to 32x32 and normalized to zero mean with unit standard deviation. Each mini-batch has a size of 100 and is sampled from a random 20 characters to make sure the amount of similar pairs is reasonable. After pair enumeration, there are 10000 pairs subject to the loss function, which is a two-class cross entropy loss. The ground-truth similarity is obtained by converting the categorical labels. The loss of G is optimized by stochastic gradient descent and is the only part trained in a supervised manner in this section.

The Constrained Clustering Network (CCN) is used to reconstruct the semantic clusters in the target dataset using outputs from G. The network has four 3x3 convolution layers followed by 2x2 max-pooling with stride 2, one hidden fully-connected layer, and one cluster assignment layer which is also fully connected. The number of output nodes in the last layer is equal to the number of potential clusters in the target data. The output distribution is enumerated in pairs before sending to LCO. The network is randomly initialized, trained

end-to-end, and optimized by stochastic gradient descent with randomly sampled 100 images per mini-batch. Note the G function used by LCO is fixed during the optimization. The input data preparation is the same as above, except now the data is from $Omniglot_{eval}$. Specifically, during the training, the same mini-batch is given to both G and CCN. The dense pairwise similarity predictions from G are sent to LCO and are then fully utilized. The only hyper-parameter in LCO is σ, and we set it to 2 for all our experiments.

$Omniglot_{eval}$ contains 20 alphabets and each one can be used as a standalone dataset. The 20 target datasets contain a varied number (20 to 47) of characters. Therefore we can evaluate the reconstruction performance under a varied number of ground-truth clusters. We tested the situations when the number of character (K) in an alphabet is known and unknown. When K is known, we set the target number of clusters in the clustering algorithm equal to the true number of characters. If K is unknown, the common practice is to set it to a large number so that the data from different categories will not be forced to be in the same cluster. In the experiment, we merely set K to 100, which is much larger than the largest dataset (47).

All constrained clustering algorithms can be used to reconstruct the semantic clusters for our problem. Since there are mispredictions in G, robustness to noise is the most important factor. Here we include all four types of constrained clustering algorithms introduced in Chapter 2 as the baselines. We provide the full set of pairwise constraints to all algorithms including ours. In other words, given an alphabet of 20 characters, it contains 400 images and 160000 predicted similarities from G (note that G makes predictions for both orders of a pair and for self-pairs). The full pairwise similarities were presented to all algorithms in random order, while we empirically found it has no noticeable effect on results. We pick the baseline approaches based on code availability and scalability concerning the number of pairwise constraints. Therefore we have shown results for K-means [23], LPNMF [25], LSC [24], ITML [15], SKKm [16], SKLR [17], SKMS [16], CSP [19] , and MPCK-means [21] as our baselines. We use the default parameters for each algorithm

Table 4.3: Unsupervised cross-task transfer from $Omniglot_{bg}$ to $Omniglot_{eval}$. The performance is averaged across 20 alphabets which have 20 to 47 letters. The ACC and NMI without brackets have the number of clusters equal to ground-truth. The "(100)" means the algorithms use $K = 100$. The characteristics of how each algorithm utilizes the pairwise constraints are marked in the "Constraints in" column, where metric stands for the metric learning of feature representation. The expanded version of this table is available in Appendix Table A.1.

Method	Constraints in		ACC	ACC (100)	NMI	NMI (100)
	Metric	Clustering				
K-means			21.7%	18.9%	0.353	0.464
LPNMF			22.2%	16.3%	0.372	0.498
LSC			23.6%	18.0%	0.376	0.500
ITML	o		56.7%	47.2%	0.674	0.727
SKMS	o		-	45.5%	-	0.693
SKKm	o		62.4%	46.9%	0.770	0.781
SKLR	o		66.9%	46.8%	0.791	0.760
CSP		o	62.5%	65.4%	0.812	0.812
MPCK-means	o	o	81.9%	53.9%	0.871	0.816
CCN (Ours)	o	o	**82.4%**	**78.1%**	**0.889**	**0.874**

provided by the original author except for K, the number of clusters. We use the same normalized images used in CCN for all algorithms.

The evaluation uses two clustering metrics. The first is normalized-mutual information (NMI) [107] which is widely used for clustering. The second one is the clustering accuracy (ACC) [108]. The ACC metric first finds the one-to-one matching between predicted clusters and ground-truth labels, and then calculates the classification accuracy based on the mapping. All data outside the matched clusters will be regarded as mis-predictions. To get high ACC, the algorithm has to generate coherent clusters where each cluster includes most of the data in a category; otherwise the score drops quickly. Therefore ACC provides better discrimination to evaluate whether the semantic clusters have been reconstructed well.

4.5.1.2 Results and Discussions

We report the average performance over the 20 alphabets in table 4.3. Our approach achieved the top performance on both metrics. The CCN demonstrates strong robustness on

Table 4.4: The performance of the similarity prediction function used in section 4.5.1. We leverage the N-way test which is commonly used in one-shot learning evaluation. The similarity is learned with $Omniglot_{bg}$ and has N-way test with $Omniglot_{eval}$ and MNIST. The experimental settings follow [109]. The raw probability output (without binarization) from our G is used to find the nearest exemplar in the N-way test.

	Omniglot-eval		MNIST
Method	5-way	20-way	10-way
Siamese-Nets [110]	0.967	0.880	0.703
Match-Net [109]	**0.981**	**0.938**	**0.720**
Ours	0.979	0.935	**0.720**

the challenging scenario of unknown K. It achieved 78.1% average accuracy. Compared with 82.4% when K is known, CCN has a relatively small drop. Compared to the second best algorithm, CSP, which is 65.4%, CCN outperforms it with a large gap. The classical approach MPCK-means works surprisingly well when the number of clusters is known, but its performance dropped dramatically from 81.9% to 53.9% when $K = 100$. In the performance breakdown for the 20 individual alphabets, CCN achieved 94% clustering accuracy on *Old Church Slavonic Cyrillic*, which has 45 characters, Therefore the results show the feasibility of reconstructing semantic clusters using only noisy similarity predictions.

Where to inject the pairwise similarity? The experiments in table 4.3 show a clear trend that utilizing the pairwise constraints jointly for both metric learning and minimizing the clustering loss achieves the best performance, including both MPCK-means and CCN. In the case of unknown number of clusters, where we set $K = 100$, the algorithms that use constraints to optimize clustering loss have better robustness, for example, CSP and CCN. The group that only use constraints for metric learning (ITML, SKMS, SKKm, and SKLR) significantly outperform the group that does not use it (K-means, LPNMF, LSC). However, their performance are still far behind CCN. Our results confirm the importance of jointly optimizing the metric and clustering.

The robustness against noisy similarity prediction is the key factor to enable the cross-task transfer framework. To the best of our knowledge, table 4.3 is the first comprehensive

robustness comparisons using predicted constraints learned from real data instead of converting from ground-truth labels. The accuracy of G in our experiment is shown in Table 4.4 and demonstrates the reasonable performance of G which is on par with Matching-Net [109]. After binarizing the prediction at 0.5 probability, the similar pair precision, similar pair recall, dissimilar pair precision, and dissimilar pair recall among the 659 characters are $(0.392, 0.927, 0.999, 0.995)$, accordingly. The binarized predictions are better than uniform random guess $(0.002, 0.500, 0.998, 0.500)$, but are still noisy. Therefore it is very challenging for constrained clustering. The robustness of CCN is significantly stronger than other methods. We hypothesize that during the optimization, the gradients from wrongly predicted pairs are canceled out by each other or by the correctly predicted pairs. Therefore the overall gradient still moves the solution towards a better clustering result.

How to predict K? Inferring the number of clusters (NC) is a hard problem, but with the pairwise similarity information, it becomes feasible. For evaluation, we compute the difference between the number of dominant clusters (NDC) and the true number of categories (NC^{gt}) in a dataset. We use a naive definition for NDC, which is the number of clusters that have a size larger than expected size when data is sampled uniformly. In other words, $NDC = \sum_{i=1}^{K} [C_i >= E[C_i]]$, where $[\cdot]$ is an Iverson Bracket and C_i is the size of cluster i. For example, $E[C_i]$ will be 10 if the alphabet has 1000 images and $K = 100$. Then the average difference ($ADif$) is calculated by $ADif = \frac{1}{|D|} \sum_{d \in D} |NDC_d - NC_d^{gt}|$, where d (i.e., alphabet) is a dataset in D. A smaller $ADif$ indicates a better estimate of K. CCN achieves a score of 6.35 (Table 4.5). We compare this with the baseline approach SKMS [16], which does not require a given K and supports a pipeline to estimate K automatically (therefore we only put it into the column $K = 100$ in table 4.3.). SKMS gets 16.3. Furthermore, 10 out of 20 datasets from CCN's prediction have a difference between NDC_d and NC_d^{gt} smaller or equal to 3, which shows the feasibility of estimating K with predicted similarity.

Table 4.5: Estimates for the number of characters across the 20 datasets in $Omniglot_{eval}$. The bold number means the prediction has error smaller or equal to 3. The $ADif$ is defined in section 4.5.1.2.

Alphabet	True #class	SKMS	CCN (K=100)
Angelic	20	16	26
Atemayar_Qelisayer	26	17	34
Atlantean	26	21	41
Aurek_Besh	26	14	**28**
Avesta	26	8	32
Ge_ez	26	18	32
Glagolitic	45	18	**45**
Gurmukhi	45	12	**43**
Kannada	41	19	**44**
Keble	26	16	**28**
Malayalam	47	12	**47**
Manipuri	40	17	**41**
Mongolian	30	**28**	36
Old_Church_Slavonic_Cyrillic	45	23	**45**
Oriya	46	22	**49**
Sylheti	28	11	50
Syriac_Serto	23	19	38
Tengwar	25	12	41
Tibetan	42	15	**42**
ULOG	26	15	40
$ADif$		16.3	6.35

Table 4.6: Unsupervised cross-task transfer learning on ImageNet. The values are the average of three random subsets in $ImageNet_{118}$. Each subset has 30 classes. The "ACC" has $K = 30$ while the "ACC (100)" sets $K = 100$. All methods use the features (outputs of average pooling) from Resnet-18 pre-trained with $ImageNet_{882}$ classification.

Method	ACC	ACC(100)	NMI	NMI(100)
K-means	71.9%	34.5%	0.713	0.671
LSC	73.3%	33.5%	0.733	0.655
LPNMF	43.0%	21.8%	0.526	0.500
CCN (ours)	**73.8%**	**65.2%**	**0.750**	**0.715**

Table 4.7: Performance of the similarity prediction function (G, trained with $ImageNet_{882}$) applied to three subsets of $ImageNet_{118}$. Each subset contains 30 random classes of $ImageNet_{118}$. The predictions are binarized at 0.5 to calculate the precision and recall. Random*: The expected performance when classes are uniformly distributed and make uniform random guess for similarity. It is an approximation since the number of images in each class is only roughly equal in ImageNet. We sampled 12M pairs for each set to collect the statistics. Sampling more pairs has no noticeable change to the values.

Set	Similar Precision	Similar Recall	Dissimilar Precision	Dissimilar Recall
Random*	0.033	0.500	0.967	0.500
Set A	0.840	0.664	0.983	0.994
Set B	0.825	0.631	0.981	0.993
Set C	0.771	0.671	0.983	0.990
Average	0.812	0.655	0.982	0.992

4.5.1.3 Experiments using the ImageNet dataset

To demonstrate the scalability of our approach, we applied the same scheme on the ImageNet dataset. The 1000-class dataset is separated into 882-class ($ImageNet_{882}$) and 118-class ($ImageNet_{118}$) subsets as the random split in [109]. We use $ImageNet_{882}$ for A and 30 classes ($\sim$39k images) are randomly sampled from $ImageNet_{118}$ for T. The difference from section 4.5.1.1 is that here we use Resnet-18 for both G and CCN, and the weights of the backbone are pre-trained with $ImageNet_{882}$. Since the number of pairs is high and it is not feasible to feed them into other constrained clustering algorithms, we compare CCN with K-means, LSC[24], and LPNMF [25]. We use the output from the av-

erage pooling layer of Resnet-18 as the input to these clustering algorithms. CCN gives the top performance with average ACC 73.8% when K is known, and 65.2% when the number of clusters is unknown, which outperforms the second (34.5% by K-means) with a large margin. The full comparison is in Table 4.6. And the performance of G is shown in Table 4.7.

4.5.2 Unsupervised Cross-Domain Transfer Learning

4.5.2.1 Setup

Office-31 [111] has images from 31 categories of office objects. The 4652 images are obtained from three domains: Amazon (a), DSLR (d), and Webcam (w). The dataset is the standard benchmark for evaluating domain adaptation performance in computer vision. We experiment with all six combinations (source S' $\rightarrow$ target T): $a \rightarrow w$, $a \rightarrow d$, $d \rightarrow a$, $d \rightarrow w$, $w \rightarrow a$, $w \rightarrow d$, and report the average accuracy based on five random experiments for each setting.

The G function learns the semantic similarity function from the auxiliary dataset A, which is ImageNet with all 1000 categories. The backbone network of G is Resnet-18 and has the weights initialized by ImageNet classification. The training process is the same as section 4.5.1.1 except the images are resized to 224.

We follow the standard protocols using deep neural networks [43, 38] for unsupervised domain adaptation. The backbone network of CCN^{+} is pre-trained with ImageNet. Figure 4.6 illustrates the scheme. During the training, all source data and target data are used. Each mini-batch is constructed by 32 labeled samples from source and 96 unlabeled samples from target. Since the target dataset has no labels and could only be randomly sampled, it is crucial to have sufficient mini-batch size to ensure that similar pairs are sampled. The loss function used in our approach is equation 4.3 and is optimized by stochastic gradient descent. The CCN$^{+/++}$ and DANN (RevGrad) with ResNet backbone are implemented with Torch. We use the code from original author for JAN. Both DANN and JAN use a 256-

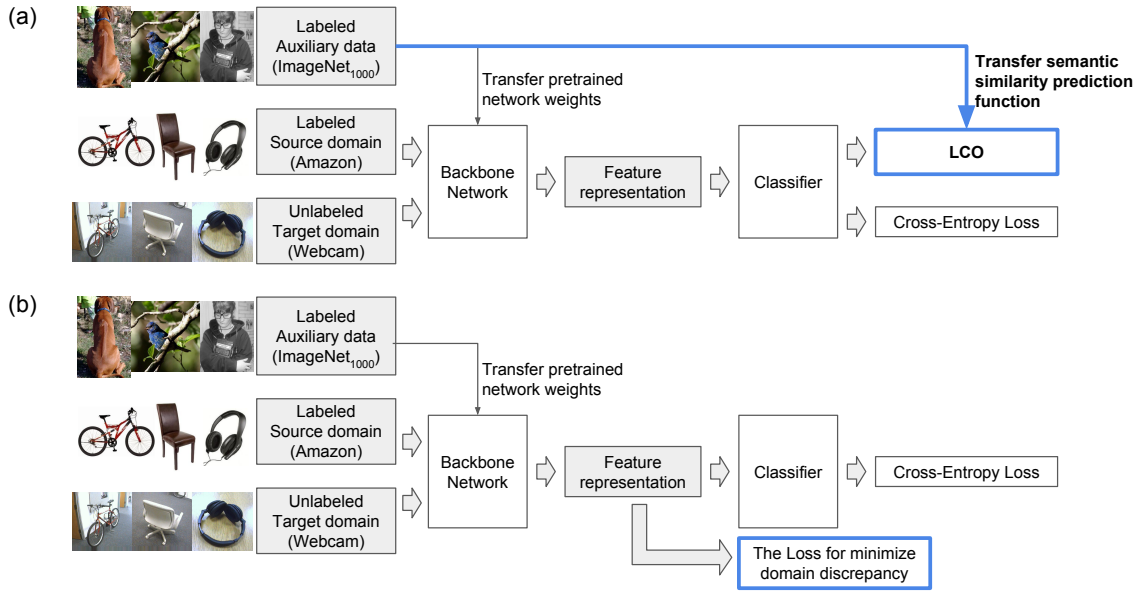


Figure 4.6: Comparison between domain adaptation approaches (a) Transferring semantic similarity from auxiliary data (our method), and (b) Minimizing the domain discrepancy. The diagram uses office-31 benchmark as the scenario of transferring. The cross-entropy loss is only applied to the labeled source data, while the criteria in blue box apply to both source and target data.

dimension bottleneck feature layer.

4.5.2.2 Results and Discussions

The results are summarized in table 4.8. Our approach (CCN$^+$) demonstrates a strong performance boost for the unsupervised cross-domain transfer problem. It reaches 77.5% average accuracy which gained 6.2 points from the 71.3% source-only baseline. Although

Table 4.8: Unsupervised cross-domain transfer (domain adaptation) on the Office-31 dataset. The backbone network used here is Resnet-18 [112] pre-trained with ImageNet.

	A → W	D → W	W → D	A → D	D → A	W → A	Avg
Source-Only	66.8	92.8	96.8	67.1	51.4	53.0	71.3
DANN [38]	73.2	97.0	99.0	69.3	58.0	57.8	75.7
JAN [43]	74.5	94.1	**99.6**	**75.9**	58.7	59.0	76.9
CCN$^+$ (ours)	76.7	97.3	98.2	71.2	61.0	60.5	77.5
CCN^{++} (with DANN)	**78.2**	**97.4**	98.6	73.5	**62.8**	**60.6**	**78.5**

Table 4.9: Performance of the similarity prediction function (G, trained with $ImageNet_{882}$) applied on three domains of the Office-31 dataset. In total, 1.4M pairs are examined to calculate the table.

Domain	Similar Precision	Similar Recall	Dissimilar Precision	Dissimilar Recall
Amazon	0.194	0.696	0.988	0.894
DSLR	0.196	0.882	0.995	0.858
Webcam	0.213	0.865	0.994	0.876

our approach merely transfers more information from the auxiliary dataset, it outperforms the strong approach DANN (75.7%), and state-of-the-art JAN (76.9%). When combining ours with DANN (CCN^{++}), the performance is further boosted. This indicates that LCO helps mitigate the transfer problem in a certain way that is orthogonal to minimizing the domain discrepancy. We observe the same trend when using a deeper backbone network, i.e., ResNet-34. In such a case the average accuracy achieved is 77.9%, 81.1% and 82% for source-only, CCN^{+} and CCN^{++}, respectively, though we used exactly the same G as before (with ResNet-18 backbone for G). This indicates that the information carried in the similarity predictions is not equivalent to transferring features with deeper networks. Results with deeper networks are in Appendix Table A.2 and the performance of G is provided in Table 4.9 to show that although the prediction has low precision for similar pairs (~ 0.2), our approach still benefits from the dense similarity predictions.

4.5.2.3 Experiments using SVHN-to-MNIST

We also evaluated the CCN^{+} on another widely compared scenario, which uses color Street View House Numbers images (SVHN) [114] as S', the gray-scale hand-written digits (MNIST) [115] as T. To learn G, we use the $Omniglot_{bg}$ as A. We train all the networks in this section from scratch. Our experimental setting is similar to [41]. We achieve the top performance with 89.1% accuracy. The performance gain from source-only in our approach is +37.1%, which wins by a large margin compared to the +23.9% of LTR [41]. The full comparison is presented in Table 4.10.

Table 4.10: Unsupervised transferring across domains (S': SVHN, T: MNIST A: $Omniglot_{bg}$) without pre-trained backbone network weights. Our setup is similar to [41] and [38] which therefore has a similar source-only performance.

Method		SVHN-¿MNIST	Gain
[38]	Source-Only	54.9	-
	DANN	73.9	+19.0
[41]	Source-Only	54.9	-
	LTR	78.8	+23.9
[113]	Source-Only	70.1	-
	ATDA	86.2	+16.1
[46]	Source-Only	60.1	-
	ADDA	76.0	+15.9
Ours	Source-Only	52.0	-
	CCN$^+$	**89.1**	**+37.1**

4.6 Conclusion and Outlook

In this chapter, we demonstrated the usefulness of transferring information in the form of pairwise similarity predictions. Such information can be transferred as a function and utilized by a loss formulation inspired from constrained clustering, but implemented more robustly within a neural network that can jointly optimize both features and clustering outputs based on these noisy predictions. The experiments for both cross-task and cross-domain transfer learning show strong benefits of using the semantic similarity predictions resulting in better results than previous works across several datasets. This is true even without explicit domain adaptation for the cross-domain task, and if we add a domain discrepancy loss the benefits increase further.

There are two key factors that determine the performance of the proposed framework. The first is the robustness of the constrained clustering and second is the performance of the similarity prediction function. We show robustness of CCN empirically, but we do not explore situations where learning the similarity function is harder. For example, such cases arise when there are a small number of categories in source or a large domain discrepancy

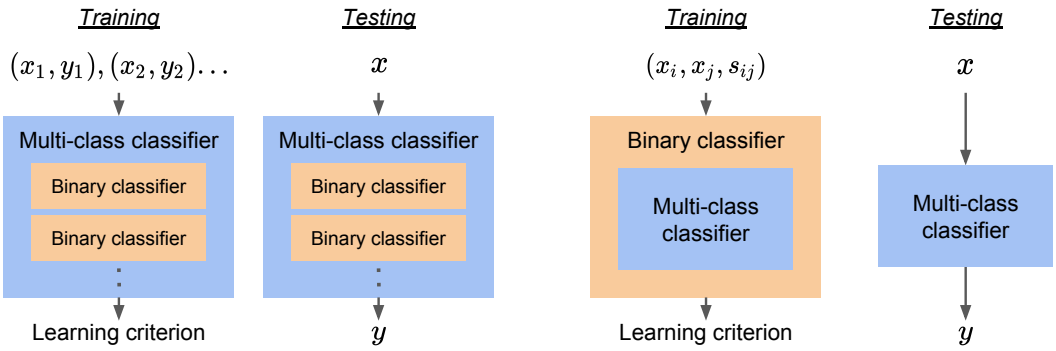
between source and target. One idea to deal with such a situation is learning G with domain adaptation strategies. We leave these aspects for future work. The work of this chapter is published in [11].

CHAPTER 5
LEARNING MULTI-CLASS CLASSIFIERS WITH ONLY PAIRWSIE INFORMATION

5.1 Introduction

In previous sections, we defined the output nodes of a neural network to clusters. The neural network is then optimized for being able to assign data to clusters through a forward propagation. The optimization and inference steps are the same as learning a standard classification task. Such resemblance between clustering and classification raises a question of whether a neural network learned with pairwise similarity for clustering is equivalent to a discriminatively trained multi-class classifier, *i.e.* $P(y|x)$ with input x and class label y.

To investigate the above question, we propose to *reduce* the problem of classification to a meta problem that underlies a set of learning problems [10]. Instead of solving the target task directly (learning a multi-class discriminative model such as a neural network), we learn a model that does not require explicit class label y but rather a weaker form of information (pairwise information). In the context of classification, the meta problem that we use is a binary decision problem, similar to previous chapters. Note that such a conversion to a different task (e.g. binary) is called a *problem reduction* method [116] which has had a long history in the literature, especially in ensemble methods and binarization techniques [117]. The most well-known strategies are "one-vs-all" [118, 119] and "one-vs-one" [120, 121, 122]. Although they have varied ensembling strategies, all of them share the same task encapsulating scheme, as illustrated in Figure 5.1a; specifically the binary classifiers are the sub-modules of a multi-class classifier (i.e. the multi-class classifier consists of multiple binary classifiers). These schemes still require that the class label y be available to create the inputs for each binary classifier, and therefore these strategies do not relax the



(a) Binary classifier ensemble (b) Meta classification learning

Figure 5.1: Problem reduction schemes for multi-class classification. This work proposes scheme (b), which introduces a binary classifier that captures s_{ij}. Note that s_{ij} represents the probability that x_i and x_j belong to the same class.

labeling requirements mentioned earlier.

In this chapter, we propose a novel strategy to address the above limitations. Our method reverses the task encapsulation order so that a multi-class classifier becomes a sub-module of a binary classifier, as illustrated in Figure 5.1b. The connection between the two classifiers is elucidated in Section 5.3. There are two highlights in Figure 5.1b. First, class labels y_i are not required in the learning stage. Instead, our method uses pairs of data (x_i, x_j) as input and pairwise similarity s_{ij} for the supervision. Second, there is only one binary classifier in the scheme and it is present only during the training stage. In other words, the ephemeral binary task assists the learning of a multi-class classifier without being involved in the inference. When using a neural network with softmax outputs for the multi-class classifier, the proposed scheme can learn a discriminative model with only pairwise information.

We specifically make the following contributions: 1) We analyze the problem setting and show that the loss we can use for this encapsulation can be easily derived, and we present an intuitive probabilistic graphical model interpretation for doing so, 2) We evaluate its performance compared to vanilla supervised learning of neural networks which uses multi-class labels, and visualize the loss landscape to better understand the underly-

ing optimization difficulty, and 3) We demonstrate support for learning classifiers in more challenging problem domains, e.g. in unsupervised cross-task transfer and semi-supervised learning. We show how our meta classification framework can support all three learning paradigms, and evaluate it against several state-of-the-art methods. The experimental results show that the same meta classification approach is superior or comparable to state of the art across the three problem domains (supervised learning, unsupervised cross-task learning, and semi-supervised learning), demonstrating flexibility to support even unknown types and numbers of classes.

5.2 Related Work

Previous works presented a unifying framework for multi-class classification by reducing it to multiple binary problems [116]. The concepts for achieving such a reduction, one-vs-all and one-vs-one, have been widely adopted and analyzed [117]. The two strategies have been used to create several popular algorithms, such as variants of support vector machine [123], AdaBoost [124, 125], and decision trees [126]. Despite the long history of reduction, our proposed scheme (Figure 5.1b) has not been explored. Furthermore, the scheme can be deployed easily by replacing the learning objective, which is fully compatible with deep neural networks for classification, a desirable property for broad applicability. Lastly, since we use the pairwise similarity for learning, this chapter will mainly compare to the constrained clustering methods discussed in Chapter 3.

Unsupervised cross-task transfer learning: We use the same setting described in Chapter 4. Our new learning objective inspired from Figure 5.1b can be a drop-in replacement for the KCL criteria used in Chapter 4.

Semi-supervised learning: Our meta classification strategy can easily plug into a semi-supervised learning scheme. Our comparison focuses on state-of-the-art methods which solely involve adding a consistency regularization [127, 128, 129, 130] or Pseudo-Labeling [131] for training a neural network. Another line of strategy combines weak supervision,

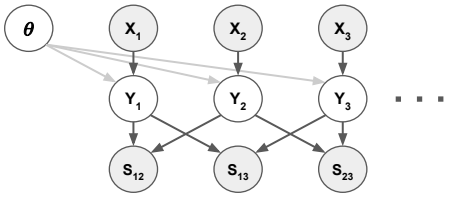


Figure 5.2: Graphical representation for the meta classification task; X_i represents the node of input data, Y_i represents the class label, S_{ij} is pairwise similarity between instances i and j, and θ represents the neural network parameters.

such as similar pairs, and unlabeled data to learn a binary classifier [132]. We present a new method by augmenting Figure 5.1b with the Pseudo-Labeling strategy.

5.3 Meta Classification Likelihood (MCL)

A natural way to analyze problems with observed and unobserved information is through a probabilistic graphical model. In figure 5.2, we show the graphical model for our problem, where class-specific labels Y are latent while pairwise similarities S are observed. Specifically, we denote $\mathbf{X} = \{X_1, .., X_n\}$, $\mathbf{Y} = \{Y_1, .., Y_n\}$, and $\mathbf{S} = \{S_{ij}\}_{1 \leq i,j \leq n}$ to represent the nodes for samples, class labels, and pairwise similarities, respectively. In the model, we have $Y_i \in \{1, 2, .., C\}$ and $S_{ij} \in \{0, 1\}$. Then we have $\mathtt{P}(S_{ij} = 1 | Y_i, Y_j) = 1$ when $Y_i = Y_j$ and zero probability otherwise; similarly, $\mathtt{P}(S_{ij} = 0 | Y_i, Y_j) = 1$ when $Y_i \neq Y_j$. The output of a discriminative classifier with parameters θ is $f(x_i; \theta) = \mathtt{P}(Y_i | x_i; \theta)$, where $f(x_i; \theta)$ outputs a categorical distribution. Now we describe the likelihood that the model explains the observed labeling (either with class labeling or pairwise labeling).

$$\mathcal{L}(\theta; \mathbf{X}, \mathbf{Y}, \mathbf{S}) = \mathtt{P}(\mathbf{X}, \mathbf{Y}, \mathbf{S}; \theta) = \mathtt{P}(\mathbf{S}|\mathbf{Y})\mathtt{P}(\mathbf{Y}|\mathbf{X}; \theta)\mathtt{P}(\mathbf{X}) \tag{5.1}$$

When $\mathbf{S}$ is fully observed while $\mathbf{Y}$ is unknown, calculating the likelihood requires marginalizing $\mathbf{Y}$ by computing $\sum_{\mathbf{Y}} \mathtt{P}(\mathbf{S}|\mathbf{Y})\mathtt{P}(\mathbf{Y}|\mathbf{X}; \theta)$, which is intractable. The pairwise term $\mathtt{P}(\mathbf{S}|\mathbf{Y}) = \prod_{i,j} \mathtt{P}(S_{ij}|Y_i, Y_j)$ makes all Y_i dependent on each other and prohibits efficient factorization. Thus, we approximate the computation by imposing additional inde-

pendences such that $S_{ij} \perp \mathbf{S} \setminus \{S_{ij}\} | X_i, X_j$ Now we can compute the likelihood with the observed nodes $X_i = x_i$ and $S_{ij} = s_{ij}$:

$$\mathcal{L}(\theta; \mathbf{X}, \mathbf{S}) \approx \sum_{\mathbf{Y}} \mathtt{P}(\mathbf{S}|\mathbf{Y}) \mathtt{P}(\mathbf{Y}|\mathbf{X}; \theta) \tag{5.2}$$

$$\approx \prod_{i,j} \Big(\sum_{Y_i = Y_j} \mathbb{1}[s_{ij} = 1] \mathtt{P}(Y_i|x_i;\theta) \mathtt{P}(Y_j|x_j;\theta) + \sum_{Y_i \neq Y_j} \mathbb{1}[s_{ij} = 0] \mathtt{P}(Y_i|x_i;\theta) \mathtt{P}(Y_j|x_j;\theta) \Big). \tag{5.3}$$

Equation (5.2) omits the $\mathtt{P}(\mathbf{X})$ since $\mathbf{X}$ are observed leaf nodes. It is straightforward to take a negative logarithm on equation 5.3 and derive a loss function:

$$L_{meta}(\theta) = -\sum_{i,j} \log \Big(\sum_{Y_i = Y_j} \mathbb{1}[s_{ij} = 1] \mathtt{P}(Y_i|x_i;\theta) \mathtt{P}(Y_j|x_j;\theta) + \sum_{Y_i \neq Y_j} \mathbb{1}[s_{ij} = 0] \mathtt{P}(Y_i|x_i;\theta) \mathtt{P}(Y_j|x_j;\theta) \Big) \tag{5.4}$$

$$= -\sum_{i,j} s_{ij} \log(f(x_i;\theta)^T f(x_j;\theta)) + (1 - s_{ij}) \log(1 - f(x_i;\theta)^T f(x_j;\theta)). \tag{5.5}$$

Then we define the function g by the probability of having the same class label, which is calculated by the inner product between two categorical distributions:

$$g(x_i, x_j, f(\cdot, \theta)) = f(x_i;\theta)^T f(x_j;\theta) = \hat{s}_{ij} \tag{5.6}$$

Here we use $\hat{s}_{ij}$ to denote the predicted similarity (as opposed to ground truth similarity s_{ij}). By plugging equation 5.6 into equation 5.5, L_{meta} has the form of a binary cross-entropy loss:

$$L_{meta} = -\sum_{i,j} s_{ij} \log \hat{s}_{ij} + (1 - s_{ij}) \log(1 - \hat{s}_{ij}). \tag{5.7}$$

In Figure 5.1b, the multi-class classifier corresponds to f while the binary classifier

corresponds to g. In other words, it is surprisingly simple to wrap a multi-class classifier by a binary classifier as described above. Since there are no learnable parameters in g, the weights optimized with the meta criterion L_{meta} are all in the neural network f. To minimize L_{meta}, $f(x_i;\theta)$ and $f(x_j;\theta)$ must output a sharply peaked distribution with the peak happening only at the same output node when $s_{ij} = 1$. In the case of $s_{ij} = 0$, the two distributions must have as little overlap as possible to minimize the loss. In the latter case, the two samples are pushed to be activated at the output nodes of different classes. Both properties of f's output distribution are typical characteristics of a classifier learned with class labels and using multi-class cross-entropy. The properties also illustrate the intuition of why minimizing L_{meta} helps f learn outputs similar to a multi-class classifier.

Lastly, because of the likelihood nature of L_{meta}, we call the learning criterion a Meta Classification Likelihood (MCL) in the rest of the paper.

5.4 Learning Paradigms

The supervision used in MCL is the pairwise labeling S. Due to its weaker form compared to class labels, we have the flexibility to collect it in a supervised, semi-supervised manner, or cross-task transfer. The last one was discussed in Chapter 4. The collection method determines the learning paradigms. In the first two learning paradigms, other methods (see Related Work Section) have also used pair-wise constraints similarly; our novelty is the derivation of our new learning objective, MCL, which can replace the other objectives. In the semi-supervised learning scenario, the proposed Pseudo-MCL is a new method. Details are elaborated below.

5.4.1 Supervised Learning

Supervised pairwise labeling can be directly collected from humans, or converted from existing class labeling by having $S = \{s_{ij}\}_{1\leq i,j\leq n}$, where $s_{ij} = 1$ if x_i and x_j belong to the same class, otherwise $s_{ij} = 0$. In our experiments, we use the latter setting to enable

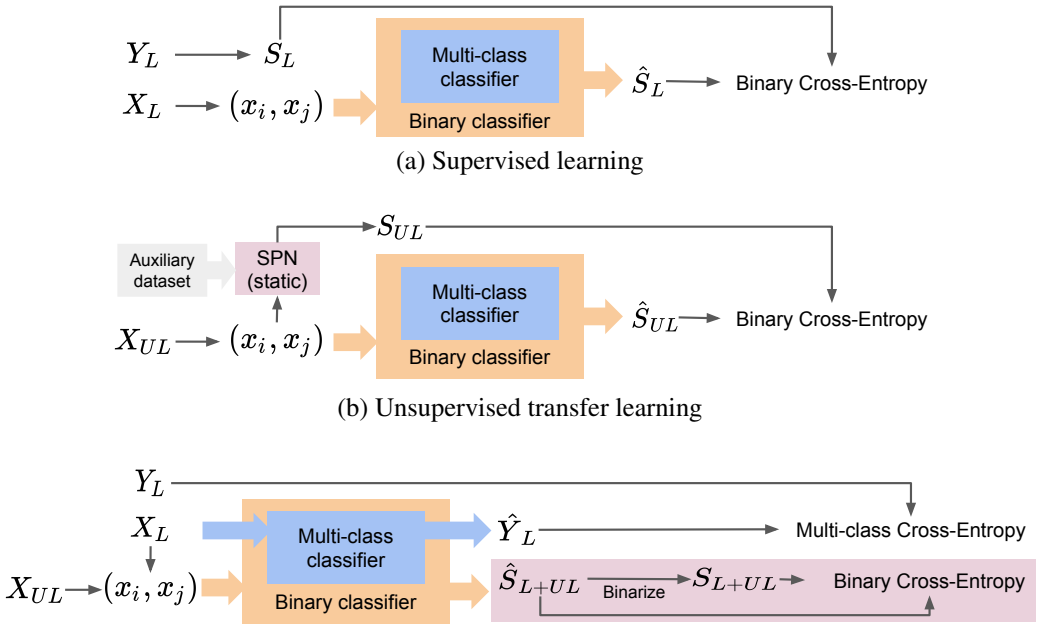


Figure 5.3: The training flows for the learning paradigms. X_L represents the labeled data with class label Y_L. X_{UL} is unlabeled data. $\hat{S}$ is the predicted pairwise similarity while S is used as the learning target. The similarity prediction network (SPN) in (b) is learned on a labeled auxiliary dataset and transferred to the target dataset X_{UL}.

comparison to other supervised algorithms. Figure 5.3a illustrates the training process.

5.4.2 Unsupervised Learning

Pairwise labeling can come from several natural cues, such as spatial and temporal proximity. For example, the patches in an image can be similar because of their spatial closeness, and the frames of video in a short time usually have similar content. Additionally, useful pairwise information can be found in the edges in social networks or in the network of academic citations. All of the above are potential applications of this work.

Another strategy that is unsupervised in the target domain is to collect pairwise labels through transfer learning. Figure 4.3 illustrated the similarity prediction network (SPN) for learning with a labeled auxiliary dataset. Then the SPN is applied on the unlabeled target dataset to predict S (the probability of being in the same class). In the last step, the predicted S is fed into a network (in the case of Chapter 4 optimized via KCL) to discover

the categories in the unlabeled target dataset. Figure 5.3b illustrates above process. Note that the classes between the auxiliary dataset and target dataset may have an overlap (cross-domain transfer) or not (cross-task transfer). In both cases, the predicted pairwise similarity is noisy (especially in the latter case); therefore the transfer learning strategy creates a challenging scenario for learning classifiers. Its difficulty makes it a good benchmark to evaluate the robustness of our methods and is used in our experiments.

5.4.3 Semi-supervised Learning

We propose a new strategy to obtain the S for semi-supervised learning. Figure 5.3c illustrates the method under the typical semi-supervised learning setting, which takes a common dataset D used for supervised learning and discards the labels for most of the dataset. The labeled and unlabeled portions in D are $D_L = (X_L, Y_L)$ and $D_{UL} = X_{UL}$ correspondingly. The main idea is to create a pseudo-similarity S_{L+UL} for the meta classifier (similar to Pseudo-Labeling [131]) by binarizing the predicted $\hat{S}_{L+UL}$ at probability 0.5. We call the method Pseudo-MCL, and we note that here interestingly g is not static as it iteratively improves as f improves. Another way to create similarity is data augmentation, inspired by the Π-model [127] or Stochastic Perturbations [128]. An image perturbed in different ways naturally belong to the same class, and thus provides free ground-truth similarity. The similarity from both methods can be easily combined to S_{L+UL} by having a logical-OR operation for the two binarized similarities. The learning objective is the sum of the multi-class cross-entropy and Pseudo-MCL, so the mapping between output nodes and classes are automatically decided by the supervised part of learning.

5.5 Experiments

5.5.1 Experimental Setup and Network Optimization

One limitation of learning a classifier without class labels is losing the mapping (the identifiability) between the output nodes and the semantic class. A simple method to obtain the

mapping is by using a part of the training data with class labels and assigning the output nodes to the dominant class which activates the node (here we obtain the optimal assignment by the Hungarian algorithm [99], which is commonly used in evaluating the clustering accuracy [108]). Note, however, that for unsupervised problems we do not need to do this except to quantitatively evaluate our method; otherwise the outputs can be seen as arbitrary clusters.

5.5.2 Supervised Learning with Weak Labels

This section empirically compares MCL to multi-class cross-entropy (CE) and the strong baseline using pairwise similarity (KCL) [133, 134]), in a supervised learning setting. Specifically, we would like to demonstrate that we can achieve similar classification rates as cross-entropy (the standard objective for multi-class classification) using only pairwise similarity, and show that the previous pairwise criterion cannot do this likely due to a poor loss landscape. We compare the classification accuracy of these criteria with varied network depths and varied dataset difficulty. The visualization of loss landscape is provided in Figure 5.4 by the method described in [135].

5.5.2.1 Quantitative analysis

We compare the classification accuracy on three image datasets: MNIST [115] is a 10-class handwritten digit dataset with 60000 images for training, and 10000 for testing; CIFAR10 and CIFAR100 [136] instances are colored 32×32 images of objects such as cat, dog, and ship. They both have 50000 images for training and 10000 for testing.

Network Architectures: We use convolution neural networks with a varied number of layers: LeNet [137] and VGG [138]. We add VGG8, which only has one convolution layer before each pooling layer, as the supplement between LeNet and VGG11. The list of architectures also includes ResNet [112] with pre-activation [139]). The number of output nodes K in the last fully connected layer is set to the true number of categories for this

Table 5.1: The classification error rate (lower is better) on three datasets with different objective functions and different neural network architectures. CE denotes that the network uses class-specific labels for training with a multi-class cross-entropy. MCL only uses the binarized similarity for learning with the meta-classification criterion. KCL is a strong baseline which also uses binarized similarity. The * symbol indicates the worst cases of KCL. The performance in parenthesis means its network uses a better initialization (VGG16 and VGG8) or a learning schedule which is 10 times longer (VGG11). The two treatments are discussed in Section 5.5.2.1. We only use VGG8 for CIFAR100 since KCL performs the best with it on CIFAR10. Each value is the average of 3 runs.

Dataset	#class	Network	(Class label) CE	(Pairwise label) KCL	MCL
MNIST	10	LeNet	0.6%	**0.5**%	0.6%
CIFAR10	10	LeNet	14.9%	16.4%	15.1%
		VGG8	10.2%	10.2%	10.2%
		VGG11	8.9%	72.2(10.4)%	9.4%
		VGG16	7.6%	*81.1(10.3)%	8.3%
		ResNet18	6.7%	73.8%	6.6%
		ResNet34	6.6%	79.3%	6.3%
		ResNet50	6.6%	79.6%	5.9%
		ResNet101	6.5%	79.9%	**5.6**%
CIFAR100	100	VGG8	**35.4**%	*45.3(40.2)%	36.1%

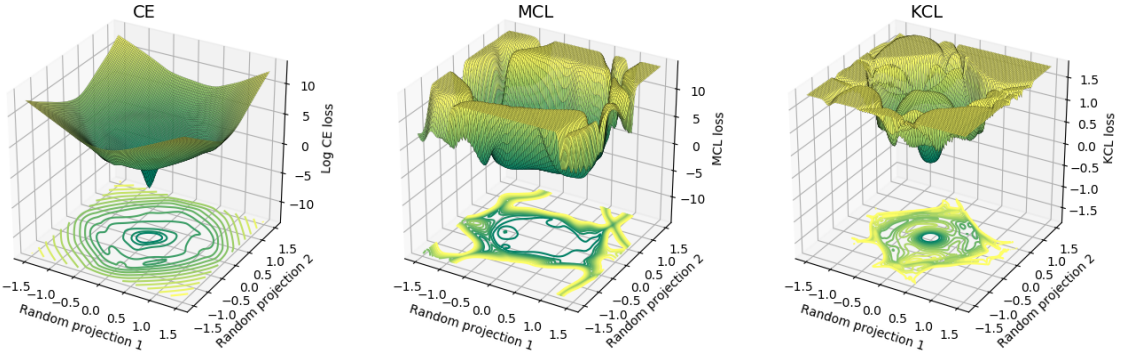
section. Since the learning objectives KCL and MCL both work on pairs of inputs, we have a pairwise enumeration layer [134] between the network outputs and the loss function.

Training Configurations: All networks in this section are trained from scratch with randomly initialized weights. By default, we use Adam [140] to optimize the three criterion with mini-batch size 100 and initial learning rate 0.001. On MNIST the learning rate was dropped every 10 epochs by a factor of 0.1 with 30 epochs in total. On CIFAR10/100 we use the same setting except that the learning rate is dropped at 80 and 120 epochs with 140 epochs in total. For CIFAR100, the mini-batch size was 1000 and the learning rate dropped at epoch 100 and 150 with 180 epochs in total. In the experiments with ResNet, we use SGD instead of Adam since SGD converges to a higher accuracy when keeping other settings the same as above. The learning rate for SGD starts with 0.1 and decays with a factor of 0.1 at the number of epochs described above.

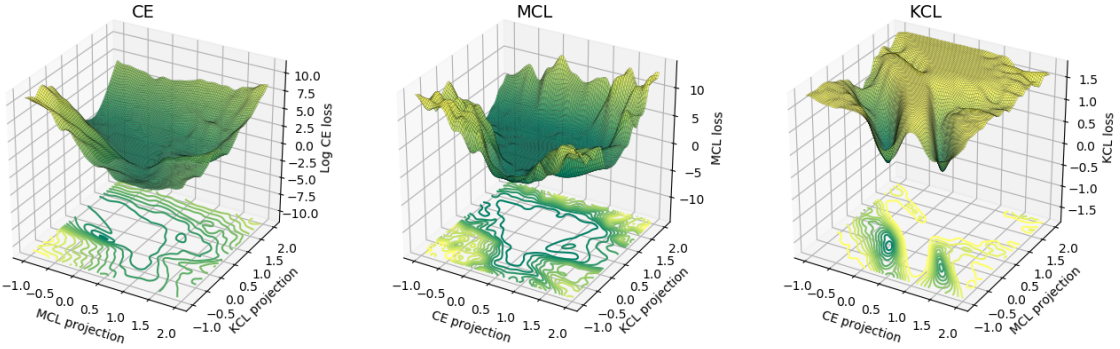
Results and discussion: The results in Table 5.1 show that MCL achieves similar classification performance as CE with different network depths and three datasets. In contrast, KCL has degenerate performance when the networks are deeper or the dataset is more difficult. This might be due to a limitation of using KL-divergence, specifically that when two probability distributions are the same, the divergence will be zero no matter what the values are. This property may introduce bad local minima or small gradients for learning. In such cases, the network initialization affects whether KCL can reach a good local minima. To investigate such a perspective, we apply two strategies. First, we use a large learning rate (0.2) with SGD to avoid bad local minima and make the training schedule 10 times longer for exploring the parameter space. This setting helps KCL with VGG11, in that the error rate drops from 72.2% to 10.4%, but not with VGG16 (from 81.1% to 76.8%). In the second strategy, we select the worst conditions (the values with * notion) in Table 5.1 for KCL and pre-train the networks with only 4k labels with CE to initialize the networks. Then we use KCL with the full training set to finish the training. With a better initialization, KCL can reach a performance close to CE and MCL. The performance is shown with parenthesis in Table 5.1. The results of both strategies indicate that KCL has bad local minima or plateaus in its loss surface (see Figure 5.4). Unlike KCL, MCL can converge to a performance close to CE with random initialization in all of our experiments. Furthermore, MCL outperforms CE with a deeper network (error rate 5.6% versus 6.5% with ResNet101). Such a result indicates that MCL is less prone to overfitting (in the Table 5.1, all ResNets achieve a training error less than 0.1%). Additional results on semi-supervised learning and cross-task transfer learning are published in [10].

5.5.2.2 Loss Landscape Visualization

We visualize the three loss functions: CE, MCL, and KCL. The loss surfaces are plotted with the function [141, 142, 135]:



(a) Random projection



(b) Mutual projection

Figure 5.4: The loss landscape visualizations. Dark green represents a low loss value while yellow means high value. The bottom part of each diagram is the 2D contour of its 3D surface. The vertical axis of CE is logarithmic to better visualize its dynamic range [135]. The MCL has a loss surface more similar to CE then KCL and has less plateau, which supports the observation that MCL converges faster than KCL.

$$f(\alpha, \beta) = L(\theta^* + \alpha\delta + \beta\eta; D) \tag{5.8}$$

where θ^* are the parameters of the model trained with loss function L and labeled dataset $D = (X, Y)$. The δ and η variables are two directions for a 2D projection of θ. The α and β are the amount of shift along δ and η from the origin θ^*. This method allows us to better understand the landscape of loss around the solution.

To choose δ and η, one straightforward method is to use random projections. However, it cannot be used to compare the geometry across different networks or loss functions, because of the scale invariance in network weights. One source of such invariance is batch normalization. In such cases, the size (i.e., norm) of a filter (assume a convolution layer) is irrelevant because the output of each layer is re-scaled during batch normalization. [135] propose Filter-wise Normalization to address the above concern. We adopt this strategy to normalize the two random projections and make the relative flatness between loss surfaces comparable. We call this a random projection method.

Another way to choose δ and η is to use solutions from different loss functions. Since we have three loss functions all able to solve the same multi-class classification problem, we can use one solution (*e.g.* θ^*_{MCL} from MCL) for the θ^* and use the remaining two solutions (*e.g.* θ^*_{CE} and θ^*_{KCL}) for the two projections (*e.g.* $\delta = \theta^*_{CE} - \theta^*_{MCL}$ and $\eta = \theta^*_{KCL} - \theta^*_{MCL}$). We call this a mutual projection method.

Visualization Setting: This section uses CIFAR10 and VGG11. We choose VGG11 because it is the smallest network that KCL cannot be optimized well with a regular learning schedule. For each learning objectives, we use the best-learned models in that error rates are less than 10.4% (see Table 5.1). The parameters of three models (θ^*_{CE}, θ^*_{MCL}, θ^*_{KCL}) are used to construct an interpolated one: $\theta = \theta^* + \alpha\delta + \beta\eta$. A 91x91 grid is used to enumerate the combinations of α and β, which are the scales for the two projected directions. The loss values associated with each (α, β) are plotted in the z-direction to form a surface for visualization. Similar to [135], the vertical axis of CE is logarithmic to better

visualize its dynamic range. For more details please refer to [135].

Results and Discussion: In the random projection (Figure 5.4a), the loss landscape with CE is similar to previous work [135] which shows a nice convexity with a not-too-deep neural network (ResNet18). The solutions of MCL and KCL are both surrounded by a plateau of high loss, but MCL has a wider concave region. The same wide concavity can be seen in the mutual projection (Figure 5.4b). This is a possible explanation for why MCL still converges to a good local minimum with a randomly initialized network. Besides, the mutual projection shows that the geometry of MCL's loss landscape is similar to CE's surface, while KCL has a sharp low-loss region only around its solutions. This might be a reason why it requires a prolonged training schedule to find a good local minimum. Overall, MCL is qualitatively more similar to CE in the visualization of loss landscape.

5.5.3 Unsupervised Cross-Task Transfer Learning

The second experiment follows the transfer learning scenario described in Section 4.5.1. This scenario has two settings. The first is when the number of output nodes K equal to the number of ground truth classes C in a dataset. This setting is the same as a multi-class classification task, except no labels (both class labels or similarity labels) are provided in the target dataset. The second setting is having an unknown C, which is closer to a clustering problem. One strategy to address the unknown C is to set a large K, and we rely on the clustering algorithm to use only a necessary number of clusters to describe the dataset while leaving the extra clusters empty.

We use constrained clustering algorithms as the baselines since they can use the pairwise inputs from a similarity prediction network (SPN) [134]. In this section, the same set of binarized pairwise similarity prediction is provided to all algorithms for a fair comparison. The metric in this section is still the classification accuracy. The mapping between output nodes and classes is calculated by the Hungarian algorithm, in which each class only matches to one output node. The unmapped output nodes are all subject to the classification

error. We also include the normalized mutual information (NMI) [107] metric. We use two datasets in the evaluation.

Omniglot [106]: This dataset has 20 images for each of 1623 different handwritten characters. The characters are from 50 different alphabets and were separated into 30 background sets ($Omniglot_{bg}$) and 20 evaluation sets ($Omniglot_{eval}$) by the dataset author. The procedure uses the $Omniglot_{bg}$ set (964 characters in total) to learn the similarity function and applies it to the cross-task transfer learning on the 20 evaluation sets (this same input is used for all compared algorithms). In this test, the backbone network for classification has four convolution layers and has weights randomly initialized. Both MCL and KCL are optimized by Adam with mini-batch size 100.

ImageNet [143]: The 1000-class dataset is separated into 882-class and 118-class subsets as the random split in [109]. The procedure uses $ImageNet_{882}$ for learning the similarity prediction function and randomly samples 30 classes ($\sim$39k images) from $ImageNet_{118}$ for the unlabeled target data. In this test, the backbone classification network is Resnet-18 and has weights initialized by classification on $ImageNet_{882}$. Both learning objectives (KCL and MCL) are optimized by SGD with mini-batch size 100.

Results and Discussion: We follow the evaluation procedure (including network architectures) used in [134], therefore the results can be directly compared. The results shown in Table 5.2 and 5.3 demonstrate a clear advantage for MCL over other methods. KCL also performs well, but MCL beats its performance with a larger gap when C is unknown (ACC with K=100). MCL also estimates the number of classes in a dataset better than KCL (Appendix Table A.3). The advantage of MCL over KCL in this section is not due to the ease of optimization, since the network is shallow in the Omniglot experiment and the network is pre-trained in the ImageNet experiment. The advantage may due to the fact that MCL is free of hyper-parameters and so performs better than KCL which uses a heuristic threshold ($\sigma = 2$) [133] for its margin.

Table 5.2: Unsupervised cross-task transfer learning on Omniglot. The performance (higher is better) is averaged across 20 alphabets (datasets), in which each has 20 to 47 letters (classes). The ACC and NMI without brackets have the number of output nodes K equal to the true number of classes in a dataset, while columns with "(K=100)" represent the case where the number of classes is unknown and a fixed $K = 100$ is used.

Method	ACC	ACC (K=100)	NMI	NMI (K=100)
K-means [23]	21.7%	18.9%	0.353	0.464
LPNMF [25]	22.2%	16.3%	0.372	0.498
LSC [24]	23.6%	18.0%	0.376	0.500
ITML [15]	56.7%	47.2%	0.674	0.727
SKKm [16]	62.4%	46.9%	0.770	0.781
SKLR [17]	66.9%	46.8%	0.791	0.760
CSP [19]	62.5%	65.4%	0.812	0.812
MPCK-means [21]	81.9%	53.9%	0.871	0.816
KCL [134]	82.4%	78.1%	0.889	0.874
MCL (ours)	**83.3%**	**80.2%**	**0.897**	**0.893**

5.5.4 Semi-supervised Learning

We evaluate the semi-supervised learning performance of the Pseudo-MCL on the standard benchmark dataset CIFAR-10. The Pseudo-MCL is compared to two state-of-the-art methods, which are VAT [129] and Π-Model [127, 128]. Our list of baselines additionally includes Pseudo-Labeling [131] and SPN-MCL since they share a similar strategy with Pseudo-MCL. The SPN-MCL uses the same strategy presented in the Section 5.4.2 for

Table 5.3: Unsupervised cross-task transfer learning on ImageNet. The values (higher is better) are the average of three random subsets in $ImageNet_{118}$. Each subset has 30 classes. The "ACC" has $K = 30$. All methods use the features (outputs of average pooling) from Resnet-18 pre-trained with $ImageNet_{882}$ classification.

Method	ACC	ACC(K=100)	NMI	NMI(K=100)
K-means	71.9%	34.5%	0.713	0.671
LSC	73.3%	33.5%	0.733	0.655
LPNMF	43.0%	21.8%	0.526	0.500
KCL	73.8%	65.2%	0.750	0.715
MCL	**74.4%**	**71.5%**	**0.762**	**0.765**

Table 5.4: Test error rates (lower is better) obtained by various semi-supervised learning approaches on CIFAR-10 with all but 4,000 labels removed. Supervised refers to using only 4,000 labeled samples from CIFAR-10 without any unlabeled data. All the methods use ResNet-18 and standard data augmentation.

Method	CIFAR10 4k labels
Supervised	$25.4 \pm 1.0\%$
Pseudo-Label	$19.8 \pm 0.7\%$
Π-model	$19.6 \pm 0.4\%$
VAT	$18.2 \pm 0.4\%$
SPN-MCL	$22.8 \pm 0.5\%$
Pseudo-MCL	$\mathbf{18.0} \pm 0.4\%$

unsupervised learning, except that the SPN is trained with only the labeled portion (*e.g.* 4k labeled data) of CIFAR10 in this section. We also note that the SPN serves as a static function to provide the similarity for optimizing the regular MCL objective.

Experiment Setting: To construct the D_L, four thousand labeled data are randomly sampled from the training set (50k images) of CIFAR10. This leaves 46k unlabeled data for D_{UL}. We use 5 random D_L/D_{UL} splits to calculate the average performance. The images are augmented by the standard procedure which includes random cropping, random horizontal flipping, and normalization to zero mean with unit variance. The model for all method is the ResNet-18 (pre-activation version, [139]), which has no dropout as in a standard model. We use Adam to optimize the objective functions of all methods. The procedure begins with learning the supervised model with only the 4k labeled data; then all other methods have a fine-tuning with D_L+D_{UL} based on the learned supervised model. The supervised model (with only 4k data) is trained with initial learning rate 0.001 and a decay with factor 0.1 at epochs 80 and 120 for a total of 140 epochs. All the semi-supervised methods are trained with initial learning rate 0.001 and have a decay with factor 0.1 at epoch 150 and 250 for a total of 300 epochs. We use a shared implementation among all methods so that the major difference between methods is the regularization term in the learning objective.

Hyperparameter Tuning: All the semi-supervised learning objectives L_{SSL} here can be represented as a weighted sum of a supervised term L_{sup} and an unsupervised regularization term L_{reg}:

$$L_{SSL} = \alpha L_{sup}(X_L, Y_L) + \beta L_{reg}(X_L \cup X_{UL}) \tag{5.9}$$

For a fair comparison, one should give the same budget for tuning the hyperparameters, such as α and β. One strategy is applying an exhaustive grid search in the hyperparameter space. Such searching requires doing cross-validation and may not be applicable when the number of labeled data is small. We adopt another strategy that gives zero tuning budget for all. We decide the α and β by natural statistics, which is the ratio between the amount of data be seen by the L_{sup} and L_{reg}. Specifically:

$$\alpha = \frac{|D_L|}{|D| + |D_L|}, \quad \beta = \frac{|D|}{|D| + |D_L|} \tag{5.10}$$

One method, VAT [129], has extra hyperparameters (*e.g.* the ϵ) in its design. In that case, we use the values decided in the original paper for this dataset.

Results and Discussion: Table 5.4 presents the comparison and shows that Pseudo-MCL is on-par with the state-of-the-art method VAT [129]. The performance difference between SPN-MCL and Pseudo-MCL clearly demonstrates the benefits of having the binary classifier and the multi-class classifier optimized together. Note that comparing our Table 5.4 and a recent review [144], we have a lower baseline performance due to a lighter regularization (no dropout) and no extra data augmentation (such as adding Gaussian noise), but the relative ranking between methods is consistent. Therefore we confirm the effectiveness of Pseudo-MCL. Lastly, Pseudo-MCL is free of hyperparameter, which is a very appealing characteristic for learning with few data.

5.6 Assumption in Meta Classification Likelihood

In section 5.3, the original likelihood (eq. 5.2) relies on an additional independence assumption to simplify its negative logarithm form to a binary cross-entropy. Such an simplification raises the question of whether equation (5.3) is over-simplified. For the supervised learning case (Section 5.4.1 with results in Section 5.5.2), where the constraints are ground truth, the global solution of our likelihood is also the solution for the original likelihood. This is because if an instance is misclassified, then it will break some pair-wise constraints in both likelihoods and no longer be optimal.

Of course, in practice, there could be two issues. First, the optimization methods for more complex models (e.g. stochastic gradient descent) may find local minima. Although it is hard to show theory for this in the general case, where local optima may be found, in such cases our visualization of the loss landscape (see Figure 5.4) provides some evidence that our method has a landscape that reduces poor local minima compared to KCL. The second potential issue is when constraints may be noisy. In such cases, for example, if the noise is high and there is a dependency structure to be leveraged, jointly optimizing across many or all constraints with the original likelihood may provide additional performance (at the expense of tractability). In practice, noisy constraints actually occur in our cross-task transfer learning experiments where our similarity prediction has significant errors (e.g. in Table 5.3 ImageNet experiments the similar pair precision, similar pair recall, dissimilar pair precision, and dissimilar pair recall are 0.812, 0.655, 0.982, and 0.992 respectively). The strong performance in terms of classification accuracy for the cross-task transfer experiments (Tables 5.2 and 5.3) shows that our simplification is robust to noise.

Overall, the fact that we have demonstrated our method on five image datasets and three application scenarios (Section 5.5.2 for supervised learning, 5.5.3 for unsupervised cross-task transfer learning, and 5.5.4 for semi-supervised learning) empirically support that the proposed likelihood can overcome these two issues. It would be interesting future work to

develop methods that can incorporate constraints jointly, however.

5.6.1 Limitation

One potential limitation of applying MCL (or KCL) happens when the number of classes or clusters in a dataset is large. In such a case, similar pairs will be hard to sample randomly. Without similar pairs presented, MCL can not find the solution for a task. If similar pairs are rare during the optimization, the learning will converge very slowly or not converge. Although this issue can be easily addressed by enlarging the size of a mini-batch (*e.g.* use batch size 1000 for the CIFAR-100 dataset), such a strategy may not scales to a larger number of classes due to memory constraints. One possible way to mitigate the problem is to pre-compute the pairwise similarity before the training. Then one can sample the data based on the similarities to ensure a reasonable amount of similar pairs is presented to each mini-batch for the training.

5.7 Conclusion

This chapter presents a new strategy to learn a multi-class classification via a binary decision problem, utilizing only pair-wise information. We formulate the problem setting via a probabilistic graphical model and derive a simple likelihood objective that can be effectively optimized via neural networks. We show how this same framework can be used for three learning paradigms: supervised learning, unsupervised cross-task transfer learning, and semi-supervised learning. Results show comparable or improved results over state of the art, especially in the challenging unsupervised cross-task setting. This demonstrates the power of using pairwise similarity as weak labels to relax the requirement of class-specific labeling. We hope the presented perspective of meta classification inspires additional approaches to learning with fewer labeled data (e.g. domain adaptation and few-shot learning) as well as application to domains where weak labels are easier to obtain.

CHAPTER 6
WHEN TO TRANSFER

6.1 Introduction

In Chapter 4, we proposed methods for cross-task transfer learning, which transfers learned pairwise similarities to discover new categories. However, the effectiveness of the transfer is largely affected by how one chooses the source and target data. Empirically, we found that when the target data are perceptually similar to the source, the predicted pairwise similarity has higher accuracy, resulting in higher performance on clustering (category discovery). Such observation leads to the question of when to transfer so that meaningful categories are discovered.

A general assumption is that when the source and target domains are related to each other, the transfer is more like to be positive [28]. Although not much works quantify such relatedness, some prior works relate the transferability to domain similarity. For example, Yosinski *et al.* [145] shows that transfer learning between two random splits (similar domains) is easier than natural/man-made object splits (dissimilar domains) in ImageNet dataset. Cui *et al.* [54] uses the Earth Movers Distance (EMD) [146] to quantify the domain similarities for the classes between the source and target. They then choose the top-k categories of the source that best cover the target to pre-train the network. We similarly quantify the domain similarity, but our goal is to investigate its relatedness to the performance of category discovery.

Rabanser *et al.* [147] has a problem setting similar to this chapter. They empirically investigate the performance of several types of two-sample tests (*e.g.* MMD) on detecting the distributional shift between datasets, concluding that a larger artificial shift (*e.g.* add random noise, rotation, or adversarial perturbation) makes the detection easier. They

further point out that the representation output from a black-box function, such as trained neural network classifiers, provides a better discernability than other dimension reduction methods, such as principal components analysis and autoencoder. In this chapter, we similarly leverage the outputs from trained neural network classifiers, but we construct a more realistic distributional shift using DomainNet datasets.

Additionally, we analyze how a discrepancy metric (*e.g.* the distance metric used in two-sample tests) can rank the datasets from different levels of domain similarity. The metric can be parametric or non-parametric. For our case, we focus on non-parametric methods to avoid injecting assumptions about a form for the underlying distributions (*e.g.* Gaussianity). There are three popular non-parametric methods: they are Kolmogorov –Smirnov (KS) test [148], Wasserstein distance [149], and maximum mean discrepancy (MMD) [150]. The usage of the KS test has been more restricted in higher dimensions due to the curse of dimensionality [151]; therefore, it is not the best fit for image data. The standard Wasserstein distance requires equal mass between the two groups of testing data [152]; thus, it is not straightforward to apply on the dataset we used, which has an unbalanced amount of samples between the domains. MMD does not have the above limitations, and has proven applicability to images since it has been widely used in image domain adaptation problems [35, 37, 153, 43]. Therefore, we use MMD to collect the observations for this chapter, and we do see a very high correlation between the MMD distance and perceptual similarity.

In summary, we create a new problem setting for analyzing how the distributional similarity between groups of data affects transfer learning (in this case, specifically for learning with pairwise similarities). We investigate this problem with MMD, and show that MMD is an effective discrepancy metric, although it has some limitations, discussed at the end.

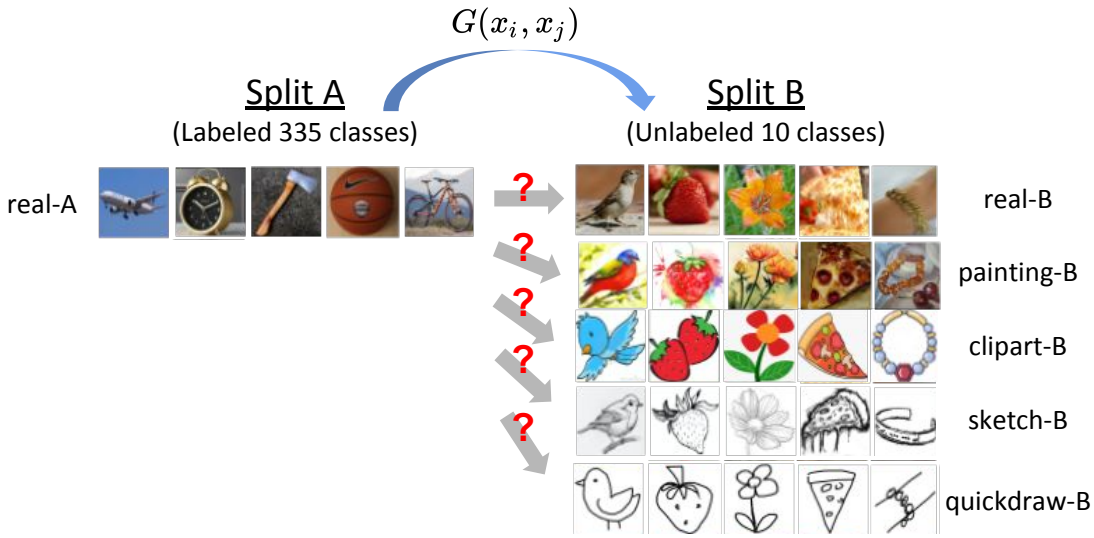


Figure 6.1: The setting for investigating "When to transfer". Split-A is the labeled source, while split-B is the unlabeled target data for category discovery. The images are from the DomainNet dataset [154].

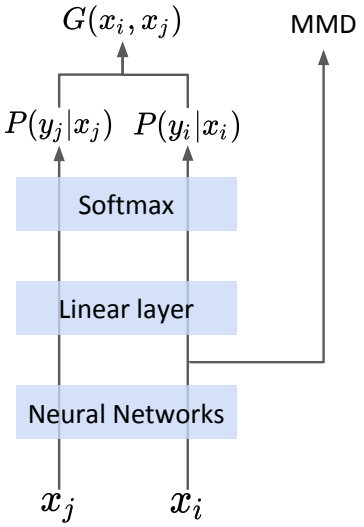


Figure 6.2: The similarity prediction function and MMD calculation are based on the outputs of a vanilla classifier.

6.2 Problem Setting

Similar to the category discovery setting used in Chapter 4.5.1, the problem setting in this chapter has a source dataset (split-A) and a target dataset (split-B), illustrated in Figure 6.1. Split A and B have exclusive sets of classes. While the split-A has the images from only the real domain, split-B has the images from multiple domains of DomainNet dataset [154]. They are real, painting, clipart, sketch, quickdraw, ordered by their perceptual similarity to the images in real-A. The protocol of transfer is the same as Chapter 4.5.1, in that the pairwise similarity prediction function is transferred from split-A to split-B, then the clustering is performed on split-B to discover the categories. This chapter investigates two questions. The first is how the perceptual domain similarity relates to discovery performance. The second is how the discovery performance relates to the MMD score. Answering the first question provides a systematic comparison for the perceptual intuition, while answering the second question provides the clue of whether the domain similarity is a good indicator for "when to transfer".

6.3 Method

The overall method for predicting pairwise similarity and calculating MMD is illustrated in Figure 6.2. Both similarity prediction function $G(x_i, x_j)$ and MMD are calculated based on the same set of feature representation, which is the outputs of the penultimate layer in a classifier $f(x)$ trained with only source data. Furthermore, the G function directly utilizes the outputs of class probability to predict the pairwise constraints. Therefore, both methods have no parameters to tune. The details of both methods are described below.

6.3.1 Maximum Mean Discrepancy

The two-sample test calculates a value to determine how different two populations are. The maximum mean discrepancy (MMD) [150] is used in such tests and is effective in estimat-

ing the visual domain similarity [54, 37, 43, 47] between two groups of data. Specifically, given two data sets, $P = \{p_1, ..., p_m\}$ and $Q = \{q_1, ..., q_n\}$, the MMD between P and Q is defined as:

$$\text{MMD}^2(P, Q) = \frac{1}{m(m-1)} \sum_{i \neq j} k(p_i, p_j) + \frac{1}{n(n-1)} \sum_{i \neq j} k(q_i, q_j) - \frac{2}{mn} \sum_{i,j} k(p_i, q_j), \tag{6.1}$$

where $k(\cdot, \cdot)$ is the Gaussian RBF kernel, i.e., $k(x, x') = \exp(-\frac{\|x-x'\|_2^2}{2\sigma^2})$. We follow the method used by [155, 59] to choose σ, where $2\sigma^2$ is set to the median distances between all image pairs in the union set $P \cup Q$.

6.3.2 From Class Prediction to Pairwise Similarity Prediction

To construct a pairwise similarity prediction function $G(x_i, x_j)$ for this chapter, we begin with having a pre-trained multi-class classifier $f(x)$, which predicts the class probability $f(x_i) = P(y_i|x_i)$. Then, the probability that a pair of samples belongs to the same class can be formulated as following:

$$P(y_i = y_j|x_i, x_j) = \sum_c P(y_i = c|x_i)P(y_j = c|x_j) \tag{6.2}$$

$$= f(x_i)^T f(x_j) \tag{6.3}$$

To generate pairwise constraints, the $G(x_i, x_j)$ binaries the probability of being a similar pair by thresholding at the expectation of $P(y_i = y_j|x_i, x_j)$. In other words:

$$G(x_i, x_j) = \begin{cases} 1, & P(y_i = y_j|x_i, x_j) \geq \mathbb{E}_{X_{train}}[P(y_i = y_j|x_i, x_j)] \\ 0, & \text{otherwise} \end{cases} \tag{6.4}$$

The expected probability is calculated with the labeled training data X_{train}. In practice, we simplify its computation by assuming a uniform class distribution in X_{train}. For example, if the X_{train} has one hundred classes, then the approximated threshold will be 0.01. The G then be used for predicting the pairwise similarity on any target datasets even they have unseen classes.

Note that the construction of G involves neither training nor any additional parameters. All it needs is a pre-trained classifier on X_{train}. That means the effectiveness of G completely relies on f, which is assumed to work well mainly when the input data comes from a distribution similar to the training distribution X_{train}.

6.4 Experiments

The overall procedure begins with a vanilla classifier $f(x)$ trained on the real-A (see Figure 6.1). The $G(x_i, x_j)$ is constructed based f and predicts pairwise similarity for MCL to perform clustering on the datasets from each domain in split-B. The number of clusters for MCL is set to 10 in this experiment. We compare the trend of similarity prediction performance and the clustering accuracy with MMD, which uses the feature representations extracted from the last hidden layer of f. For calculating the MMD with equation 6.1, the P is from the real-A, while the Q contains one of the datasets from the five domains, *i.e.* real-B, painting-B, clipart-B, sketch-B, and quickdraw-B.

Dataset: We use DomainNet [154] to create a problem setting for this chapter. DomainNet is a dataset with high-resolution images in 345 classes from 6 different domains. We include five domains in the experiments. They are real, painting, clipart, sketch, and quickdraw. The inforgrpah domain is excluded due to its noisy contents (multiple objects from different classes in a single image). We split DomainNet into two sets. The split-A has category indices 0 to 334 from the real domain, while the split-B has category indices 335 to 344 from the five domains. The split-A and split-B are used as the labeled source dataset and unlabeled target data, correspondingly. Figure 6.1 illustrates the setting.

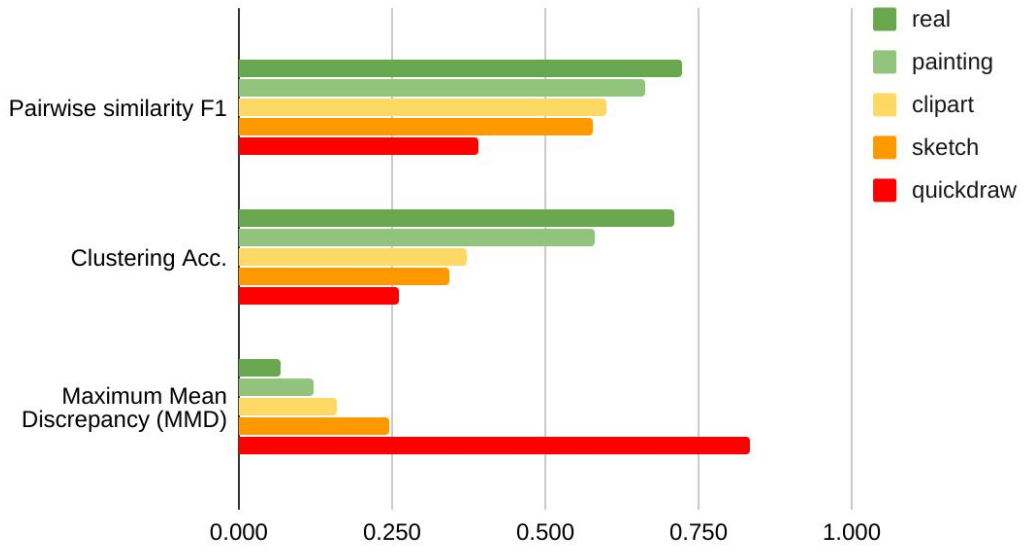


Figure 6.3: The comparison of pairwise similarity prediction, clustering accuracy, and MMD with the images from different domains. The Pearson correlation coefficient between similarity predictions and MMD is -0.96. The coefficient between clustering acc and MMD is -0.72.

Networks and Training Details: This chapter uses ResNet-34 [139] for the classifier. The ResNet-34 is trained with batch size 128 for 200 epochs with weight decay 0.0005. The optimizer is SGD with momentum 0.9, and the learning rate starts with 0.01 and decreases by factor 0.1 at 50% and 75% of the training epochs.

Evaluation Metrics: The pairwise similarity prediction performance is evaluated by its F1-score, which is the harmonic mean of precision and recall. We separately calculate the F1-score for similar and dissimilar pairs, then average the two scores. The clustering accuracy here is the same as Chapter 4.5.1.1.

Results: Figure 6.3 shows that the trend of similarity prediction is the same as the clustering accuracy. Such a trend follows the intuition of human visual perception in that the domain visually similar to real, such as painting, has a better performance. The resulting MMD negatively correlates to both the similarity prediction (Pearson correlation coefficient $\rho = -0.96$) and clustering performance ($\rho = -0.72$), showing that MMD is a good indicator for judging when to transfer the $G(x_i, x_j)$ for a reliable category discovery.

6.5 Limitation of Maximum Mean Discrepancy

The effectiveness of MMD comes with three limitations, which is also shared by the discrepancy metrics in other two-sample tests. First, the calculation needs both the source and target datasets to be available. Second, the data have to be presented in groups because the statistic of data distribution is not available or unreliable when there is only one or a few data. Third, it requires the data of different domains to be split into separate groups to allow the trend in the previous section obtainable.

The three limitations are barriers for applying MMD to a more realistic scenario. Consider a case that an intelligent agent is exploring the environment to discover new categories. The agent might not be able to keep the source dataset because of storage limitation, neither will the agent know the domain of each in-coming data. Both conditions prohibit an agent from using MMD for selecting a subset of new data to transfer and discover.

The limitations of MMD need to be relaxed. An ideal method for scoring the domain similarity uses only one target data, while no source data is required. This requirement leaves no statistics information for two-sample tests (MMD), prohibiting such a line of methods from being applicable. Therefore, in the next chapter, we investigate methods in the setting of out-of-distribution (OoD) detection, which scores domain similarity with only one testing data.

6.6 Conclusion

This chapter provides a new experimental design to confirm the intuition that perceptually similar domains transfer better than dissimilar domains, in which transferability is measured by the pairwise similarity prediction performance and its downstream clustering accuracy. The results provide empirical evidence that MMD is a good indicator to estimate domain similarity, although MMD has several limitations for application. In the next chapter, we propose methods to relax the limitations.

CHAPTER 7
SCORING THE OUT-OF-DISTRIBUTION DATA

7.1 Introduction

Out-of-distribution (OoD) detection methods assign a score for single testing data to indicate whether it comes from a distribution different from the training data. We leverage the problem setting of OoD detection, investigating existing methods, and proposing new methods for estimating the domain or distributional similarity with fewer limitations discussed in the last chapter, illustrated in Figure 7.1. This chapter focuses on the strategies which involve only mild modifications to existing discriminative classification models, published in [12] and elaborated below.

State-of-the-art machine learning models, specifically deep neural networks, are generally designed for a static and closed world. The models are trained under the assumption that the input distribution at test time will be the same as the training distribution. In the real world, however, data distributions shift over time in a complex, dynamic manner. Even worse, new concepts (*e.g.* new categories of objects) can be presented to the model at any time. Such within-class distribution shift and unseen concepts both may lead to catastrophic failures since the model still attempts to make predictions based on its closed-world assumption. These failures are therefore often silent in that they do not result in explicit errors in the model.

The above issue had been formulated as a problem of detecting whether an input data is from in-distribution (*i.e.* the training distribution) or out-of-distribution (*i.e.* a distribution different from the training distribution) [58]. This problem has been studied for many years [156] and has been discussed in several views such as rejection [157, 158], anomaly detection [159], open set recognition [60], and uncertainty estimation [160, 161, 162]. In

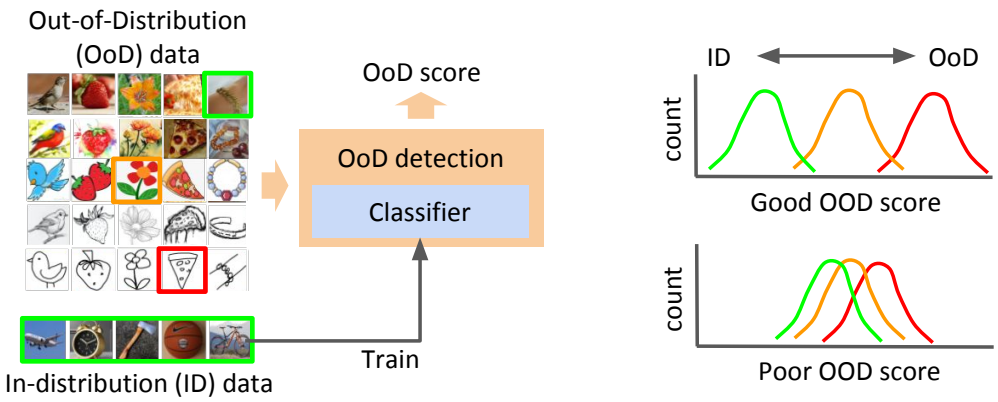


Figure 7.1: The overview of OoD detection scheme for this chapter. The left figure illustrates that the OoD detection methods are built based on the outputs (class probabilities or hidden features) of a discriminative classifier to compute an OoD score. A desired OoD scoring function results in a small overlap between ID data and OoD data for the score distribution, depicted in the right figures. Such overlap is measured by the area under the receiver operating characteristic curve (AUROC), which is extensively used in the evaluation of this chapter. The green lines/boxes represent the ID data, while orange represents a moderately distant distribution and red means a significantly distant distribution.

recent years, a popular neural network-based baseline is to use the max value of class posterior probabilities output from a softmax classifier, which can in some cases be a good indicator for distinguishing in-distribution and out-of-distribution inputs [58].

ODIN [59], based on a trained neural network classifier, provides two strategies, temperature scaling and input preprocessing, to make the max class probability a more effective score for detecting OoD data. Its performance has been further confirmed by [163], where 15 OoD detection methods are compared with a less biased evaluation protocol. ODIN out-performs popular strategies such as MC-Dropout [164], DeepEnsemble [165], PixelCNN++ [166], and OpenMax [61].

Despite its effectiveness, ODIN has a requirement that it needs OoD data to tune hyperparameters for both its strategies, leading to a concern that hyperparameters tuned with one out-of-distribution dataset might not generalize to others, discussed in [163]. In fact, other neural network-based methods [167, 168], which follow the same problem setting, have

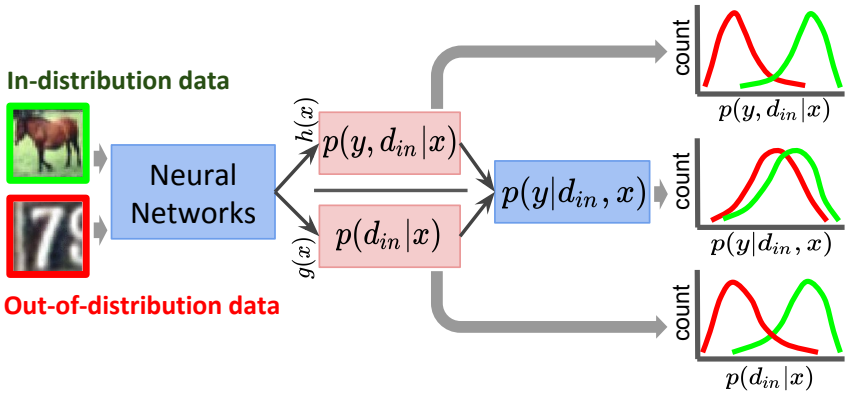


Figure 7.2: The concept of detecting out-of-distribution images by encouraging neural networks to output scores, $h(x)$ and $g(x)$, to behave like the decomposed factors in the conditional probability when the close-world assumption d_{in} is explicitly considered. Its elucidation is in Section 7.3.1. A small overlap between the green and red histograms means the x-axis a good scoring function for distinguishing OoD data from in-distribution.

a similar requirement. [169, 170] push the idea of utilizing OoD data further by using a carefully chosen OoD dataset to regularize the learning of class posteriors so that OoD data have much lower confidence than in-distribution. Lastly, [171] uses a generative model to generate out-of-distribution data around the boundary of the in-distribution for learning.

Although the above works show that learning with OoD data is effective, the space of OoD data (ex: image pixel space) is usually too large to be covered, potentially causing a selection bias for the learning. Some previous works have done a similar attempt to learn without OoD data, such as [172], which uses word embeddings for extra supervision, and [173] which applies metric learning criteria. However, both works report performance similar to ODIN, showing that learning without OoD data is a challenging setting.

In this work, we closely follow the setting of ODIN, proposing two corresponding strategies for the problem of learning without OoD data. First, we provide a new probabilistic perspective for decomposing confidence of predicted class probabilities. We specifically add a variable for explicitly adopting the closed world assumption, representing whether the data is in-distribution or not, and discuss its role in a decomposed conditional probability. Inspired by the probabilistic view, we use a dividend/divisor structure for a classifier, which encourages neural networks to behave similarly to the decomposed confidence effect. The

concept is illustrated in Figure 7.2, and we note the dividend/divisor structure is closely related to temperature scaling except that the scale depends on the input instead of a tuned hyperparameter. Second, we build on the input preprocessing method from ODIN [59] and develop an effective strategy to tune its perturbation magnitude (which is a hyperparameter of the preprocessing method) with only in-distribution data.

We then perform extensive evaluations on benchmark image datasets such as CIFAR10, CIFAR100, TinyImageNet, LSUN, SVHN, as well as a larger scale dataset DomainNet, for investigating the conditions under which the proposed strategies do or do not work. The results show that the two strategies can significantly improve upon ODIN, achieving a performance close to, and in some cases surpassing, state-of-the-art methods [167] which use out-of-distribution data for tuning. Lastly, our systematical evaluation with DomainNet reveals the relative difficulties between two types of distribution shift: semantic shift and non-semantic shift, which are defined by whether a shift is related to the inclusion of new semantic categories.

In summary, the contribution of this paper is three-fold:

- A new perspective of decomposed confidence for motivating a set of classifier designs that consider the closed-world assumption.
- A modified input preprocessing method without tuning on OoD data.
- Comprehensive analysis with experiments under the setting of learning without OoD data.

7.2 Background

This work considers the OoD detection setting in classification problems. We begin with a dataset $D_{in} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, denoting in-distribution data $\boldsymbol{x}_i \in \mathbb{R}^k$ and categorical label $y_i \in \{\mathrm{y}\} = \{1..C\}$ for C classes. D_{in} is generated by sampling from a distribution

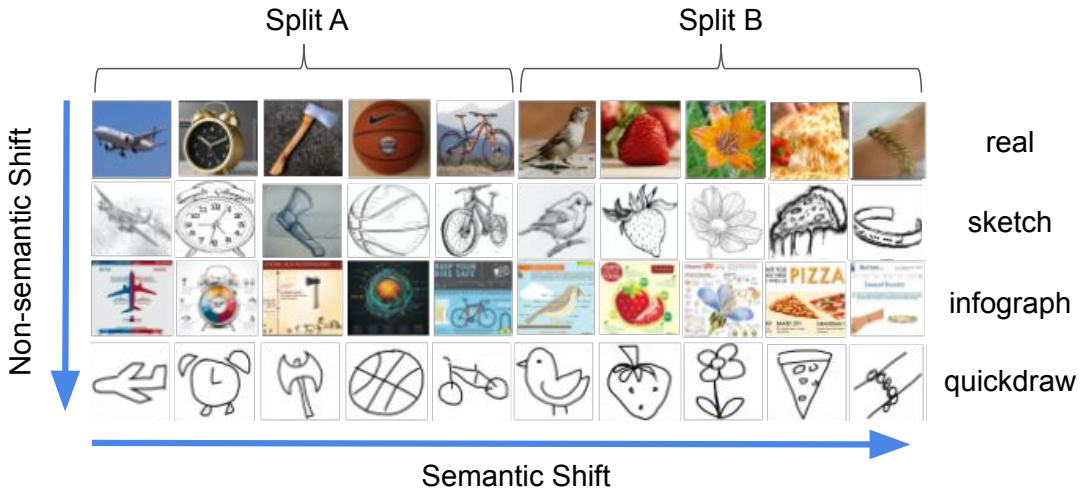


Figure 7.3: An example scheme of semantic shift and non-semantic shift. It is illustrated with DomainNet [154] images. The setting with two splits (A and B) will be used in our experiments, where only real-A is the in-distribution data.

$p_{in}(\mathbf{x}, \mathrm{y})$. We then have a discriminative model $f_\theta(\boldsymbol{x})$ with parameters θ learned with the in-domain dataset D_{in}, predicting the class posterior probability $p(\mathrm{y}|\boldsymbol{x})$.

When the learned classifier f_θ is deployed in the open world, it may encounter data drawn from a different distribution p_{out} such that $p_{out} \neq p_{in}$. Sampling from all possible distributions p_{out} that might be encountered is generally intractable especially when the dimension k is large, such as in the cases of image data. Note also that we can conceptually categorize the type of differences into non-semantic shift and semantic shift. Data with non-semantic shift is drawn from the distribution $p_{out}(\mathbf{x}, \mathrm{y})$. Examples with this shift come from the same object class but are presented in different forms, such as cartoon or sketch images. Such shift is also a scenario be widely discussed in the problem of domain adaptation [174, 154]. In the case of semantic shift, the data is drawn from a distribution $p_{out}(\mathrm{x}, \bar{\mathrm{y}})$ with $\{\bar{\mathrm{y}}\} \cap \{\mathrm{y}\} = \emptyset$. In other words, the data is from a class not seen in the training set D_{in}. Figure 7.3 has an illustration.

The above separation leads to two natural questions that must be answered for a model to work in an open world: How can the model avoid making a prediction when encountering

an input $\boldsymbol{x} \sim p_{out}(\mathrm{x}, \bar{\mathrm{y}})$, or reject a low confidence prediction when $\boldsymbol{x} \sim p_{out}(\mathrm{x}, \mathrm{y})$? In this work, we propose to introduce an *explicit* binary domain variable $\mathrm{d} \in \{d_{in}, d_{out}\}$ in order to represent this decision, with d_{in} meaning that the input is $\boldsymbol{x} \sim p_{in}$ while d_{out} meaning $\boldsymbol{x} \nsim p_{in}$ (or equivalently $\boldsymbol{x} \sim p_{out}$). Note that while generally the model cannot distinguish between the two cases we defined, we can still show that both of the questions above can be answered by estimating this single variable d.

The ultimate goal, then, is to find a scoring function $S(\boldsymbol{x})$ which correlates to the domain posterior probability $p(\mathrm{d}|\boldsymbol{x})$, in that a higher score s from $S(\boldsymbol{x})$ indicates a higher probability of $p(d_{in}|\boldsymbol{x})$. The binary decision now can be made by applying a threshold on s. Selecting such a threshold is subject to the application requirement or the performance metric calculation protocol. With the above notation, we can view the baseline method [58] as a special case with a specific scoring function $S_{Base}(\boldsymbol{x}) = \max_{\mathrm{y}} p(\mathrm{y}|\boldsymbol{x})$, where $p(\mathrm{y}|\boldsymbol{x})$ is obtained from a standard neural network classifier f_θ trained with cross-entropy loss. However, $S(\boldsymbol{x})$ can become a learnable parameterized function, and different OoD methods can then be categorized by specific parameterizations and learning procedures. A key differentiator between methods is whether the parameters are learned with or without OoD data.

7.2.1 Related Methods

This section describes the two methods that are the most related to our work: ODIN [59] and Mahalanobis [167]. These two methods will serve as strong baselines in our evaluation, especially since Mahalanobis has further been shown to have significant advantages over ODIN. Note that both ODIN and Mahalanobis start from a vanilla classifier f_θ trained on D_{in}, then have a scoring function $S(\boldsymbol{x}; f_\theta)$ which has extra parameters to be tuned. In their original work, those parameters are specifically tuned for each OoD dataset. Here we will describe methods to use them without tuning on OoD data.

ODIN comprises two strategies: temperature scaling and input preprocessing. The

temperature scaling is applied to its scoring function, which has $f_i(\boldsymbol{x})$ for the logit of i class:

$$S_{ODIN}(\boldsymbol{x}) = \max_i \frac{\exp\left(f_i(\boldsymbol{x})/T\right)}{\sum_{j=1}^{C} \exp\left(f_j(\boldsymbol{x})/T\right)} \tag{7.1}$$

Although ODIN originally involved tuning the hyperparameter T with out-of-distribution data, it was also shown that a large T value can generally be preferred, suggesting that the gain is saturated after 1000 [59]. We follow this guidance and fix $T = 1000$ in our experiments.

Mahalanobis comprises two parts as well: Mahalanobis distance calculation and input preprocessing. The score is calculated with Mahalanobis distance as follows:

$$S^{\ell}_{Maha}(\boldsymbol{x}) = \max_i -(f^{\ell}(\boldsymbol{x}) - \mu^{\ell}_i)^T \Sigma^{-1}_{\ell} (f^{\ell}(\boldsymbol{x}) - \mu^{\ell}_i) \tag{7.2}$$

$$S_{Maha}(\boldsymbol{x}) = \sum_{\ell} \alpha_{\ell} S^{\ell}_{Maha}(\boldsymbol{x}) \tag{7.3}$$

The $f^{\ell}(\boldsymbol{x})$ represents the output features at the ℓth-layer of neural networks, while μ_i and Σ are the class mean representation and the covariance matrix, correspondingly. The hyperparameter is α_{ℓ}. In the original method, α_{ℓ} is regressed with a small validation set containing both in-distribution and out-of-distribution data. Therefore they have a set of α_{ℓ} tuned for each OoD dataset. As a result, for the baseline that does not tune on OoD data we use uniform weighting $S_{Maha}(\boldsymbol{x}) = \sum_{\ell} S^{\ell}_{Maha}(\boldsymbol{x})$.

Note that both methods use the input preprocessing strategy, which has a hyperparameter to be tuned. In their original works, this hyperparameter is tuned for each OoD dataset as well. Therefore we develop a version that does not require tuning with out-of-distribution data.

7.3 Method

7.3.1 The Decomposed Confidence

[175, 176, 58] observed that the softmax classifier tends to output a highly confident prediction, reporting that "random Gaussian noise fed into an MNIST image classifier gives a predicted class probability of 91%". They attribute this to the use of the softmax function which is a smooth approximation of an indicator function, hence tending to give a spiky distribution instead of a uniform distribution over classes [58]. We acknowledge this view and further consider it as a limitation in the design of the softmax classifier. To address this limitation, our inspiration starts from reconsidering its outputs, the class posterior probability $p(\mathrm{y}|\boldsymbol{x})$, which does not consider the domain d at all. In other words, current methods condition on domain $\mathrm{d} = d_{in}$ based on the implicit closed world assumption. Thus, we use our explicit variable d_{in} in the classifier, rewriting it as the quotient of the joint class-domain probability and the domain probability using the rule of conditional probability:

$$p(\mathrm{y}|d_{in}, \boldsymbol{x}) = \frac{p(\mathrm{y}, d_{in}|\boldsymbol{x})}{p(d_{in}|\boldsymbol{x})} \tag{7.4}$$

Equation 7.4 provides a probabilistic view of why classifiers tend to be overconfident. Consider an example $\boldsymbol{x} \sim p_{out}$: It is natural to expect that the joint probability $P(\mathrm{y}, d_{in}|\boldsymbol{x})$ is low (e.g. 0.09) for its maximum value among C classes. One would also expect its domain probability $p(d_{in}|\boldsymbol{x})$ is low (e.g. 0.1). Therefore, calculating $p(\mathrm{y}|d_{in}, \boldsymbol{x})$ with Equation 7.4 gives a high probability (0.9), demonstrating how overconfidence can result. Based on the form of Equation 7.4, we call $p(\mathrm{y}, d_{in}|\boldsymbol{x})$ and $p(d_{in}|\boldsymbol{x})$ the decomposed confidence scores.

One straightforward solution for the above issue is to learn a classifier to predict the joint probability $p(\mathrm{y}, d_{in}|\boldsymbol{x})$ by having both supervision on class y and domain d. Learning to predict $p(\mathrm{y}, d_{in}|\boldsymbol{x})$ is preferred over $p(d_{in}|\boldsymbol{x})$ because it can serve both purposes for

predicting a class by $\arg\max_{y_{in}} p(\mathrm{y}, d_{in}|\boldsymbol{x})$ and rejecting a prediction by thresholding. This idea relates to the work of [170], which adds an extra loss term to penalize a predicted non-uniform class probability when an out-of-distribution data is given to the classifier. However, this strategy requires out-of-distribution data for regularizing the training.

Without having supervision on domain d (*i.e.* without out-of-distribution data), there is no principled way to learn $p(\mathrm{y}, d_{in}|\boldsymbol{x})$ and $p(d_{in}|\boldsymbol{x})$. This situation is similar to unsupervised learning (or self-supervised learning) in that we need to insert assumptions or prior knowledge about the task for learning. In our case, we use the dividend/divisor structure in Equation 7.4 as the prior knowledge to design the structure of classifiers, providing classifiers a capacity to decompose the confidence of class probability.

In the dividend/divisor structure for classifiers, we define the logit $f_i(\boldsymbol{x})$ for class i, which is the division between two functions $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$:

$$f_i(\boldsymbol{x}) = \frac{h_i(\boldsymbol{x})}{g(\boldsymbol{x})} \tag{7.5}$$

The quotient of the two scores is then normalized by the exponential function (*i.e.* softmax) for outputting a class probability $p(\mathrm{y} = i|d_{in}, \boldsymbol{x})$, which is subject to cross-entropy loss.

With the exponential normalization effect of softmax, the cross-entropy loss can be minimized in two ways: increasing $h_i(\boldsymbol{x})$ or decreasing $g(\boldsymbol{x})$. In other words, when the data is not in the high-density region of in-distribution, $h_i(\boldsymbol{x})$ may tend towards smaller values. In such case, the $g(\boldsymbol{x})$ is encouraged to be small so that the resulting logits $f_i(\boldsymbol{x})$ can further minimize the cross-entropy loss. In the other case when the data is in the high density region, $h_i(\boldsymbol{x})$ generally can reach a higher value relatively easier, thus its corresponding $g(\boldsymbol{x})$ value is less encouraged to go small. The discussed interaction between $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$ is the major driving force to encourage $h_i(\boldsymbol{x})$ to behave similar to $p(\mathrm{y} = i, d_{in}|\boldsymbol{x})$ and $g(\boldsymbol{x})$ to behave similar to $p(d_{in}|\boldsymbol{x})$, in a way that the distributional overlap between the

scores of OoD and in-distribution data is small, which is an intrinsic property of $p(y, d_{in}|\boldsymbol{x})$ and $p(d_{in}|\boldsymbol{x})$, illustrated in Figure 1.

7.3.1.1 Design Choices

Although the dividend/divisor structure provides a tendency, it does not necessarily guarantee the decomposed confidence effect to happen. The characteristic of $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$ can largely affect how likely the decomposition could happen. Therefore we discuss a set of simple design choices to investigate whether such decomposition is generally obtainable.

Specifically we have $g(\boldsymbol{x}) = \sigma(BN(\boldsymbol{w}_g f^p(\boldsymbol{x}) + b_g))$, which uses features $f^p(\boldsymbol{x})$ from the penultimate layer of neural networks sequentially through another linear layer, batch normalization (BN, optional for a faster convergence), and a sigmoid function σ. The $\boldsymbol{w}$ and b represent the learnable weights. For $h_i(\boldsymbol{x})$, we investigate three similarity measures, including inner-product (I), negative Euclidean distance (E), and cosine similarity (C) for $h_i^I(\boldsymbol{x})$, $h_i^E(\boldsymbol{x})$, and $h_i^C(\boldsymbol{x})$, correspondingly:

$$h_i^I(\boldsymbol{x}) = \boldsymbol{w}_i^T f^p(\boldsymbol{x}) + b_i; \tag{7.6}$$

$$h_i^E(\boldsymbol{x}) = -\|f^p(\boldsymbol{x}) - \boldsymbol{w}_i\|^2; \tag{7.7}$$

$$h_i^C(\boldsymbol{x}) = \frac{\boldsymbol{w}_i^T f^p(\boldsymbol{x})}{\|\boldsymbol{w}_i\|\|f^p(\boldsymbol{x})\|} \tag{7.8}$$

The overall neural network model f_θ therefore has two branches (h_i and g) after its penultimate layer (See Figure 7.2). At training time, the model calculates the logit f_i followed by the softmax function with cross-entropy loss on top of it. At testing time, the class prediction can be made by either calculating $\arg\max_i f_i(\boldsymbol{x})$ or $\arg\max_i h_i(\boldsymbol{x})$ (both will give the same predictions). For out-of-distribution detection, we use the scoring function $S_{DeConf}(\boldsymbol{x}) = \max_i h_i(\boldsymbol{x})$ or $g(\boldsymbol{x})$.

Note that when $h_i(\boldsymbol{x}) = h_i^I(\boldsymbol{x})$ and $g(\boldsymbol{x}) = 1$, this method reduces to the baseline [58]. We call the three variants of our method DeConf-I, DeConf-E, and DeConf-C. For

simplicity, the above names represent using $h_i(\boldsymbol{x})$ for the scores. The use of $g(\boldsymbol{x})$ will be indicated specifically.

7.3.1.2 Temperature Scaling

The $g(x)$ in Equation 7.5 can be immediately viewed as a learned temperature scaling function discussed in [177] and a concurrent report [178]. However, our experiment results strongly suggest that $g(x)$ is more than a scale. The $g(x)$ achieves an OoD detection performance significantly better than baselines in many experiments, indicating its potential in estimating the $p(d_{in}|\boldsymbol{x})$. More importantly, the temperature scaling is generally used as a numerical trick for learning a better embedding [179], softening the prediction [180], or calibrating the confidence [181]. Our work provides a probabilistic view for its effect, indicating such temperature might relate to how strong a classifier assumes a closed world as a prior.

7.3.2 A Modified Input Preprocessing Strategy

This section describes a modified version of the input preprocessing method proposed in ODIN [59]. The primary purpose of the modification is making the search of the perturbation magnitude ϵ to not rely on out-of-distribution data. The perturbation of input is given by:

$$\hat{\boldsymbol{x}} = \boldsymbol{x} - \epsilon \text{sign}(-\nabla_{\boldsymbol{x}} S(\boldsymbol{x})) \tag{7.9}$$

In the original method [59] the best value of ϵ is searched with a half-half mixed validation dataset of $D_{in}^{val} \sim p_{in}$ and $D_{out}^{val} \sim p_{out}$ over a list of 21 values. The perturbed images $\hat{\boldsymbol{x}}$ are fed into the classification model f_θ for calculating the score $S(\boldsymbol{x})$. The performance of each magnitude is evaluated with the benchmark metric (TNR@TPR95, described later) and the best one is selected. This process repeats for each out-of-distribution dataset, and therefore the original method results in a number of ϵ values equal to the number of out-of-distribution datasets in the benchmark.

In our method, we search for the ϵ^* which maximizes the score $S(\boldsymbol{x})$ with only the in-distribution validation dataset D_{in}^{val}:

$$\epsilon^* = \arg\max_{\epsilon} \sum_{x \in D_{in}^{val}} S(\hat{\boldsymbol{x}}) \tag{7.10}$$

Our searching criteria is still based on the same observation made by [59]. They observe that the in-distribution images tend to have their score s increased more than the out-of-distribution images when the input perturbation is applied. We therefore use Eq. 7.10 since we argue that an ϵ which makes a large score increase for in-distribution data should be sufficient to create a distinction in score. Our method also does not even require class labels although it is available in D_{in}^{val}. More importantly, our method selects only one ϵ based on D_{in}^{val} without access to the benchmark performance metric (*e.g.* TNR@TPR95), greatly avoiding the hyperparameter from fitting to a specific benchmark score. Lastly, we perform the search of ϵ on a much coarser grid, which has only 6 values: $[0.0025, 0.005, 0.01, 0.02, 0.04, 0.08]$. Therefore, our search is much faster. Although overshooting is possible (*e.g.* the maximum value is at the middle of two scales in the grid) due to the coarser grid, it can be mitigated by reducing the found magnitude by one scale (*i.e.* divide it by two). This simple strategy consistently gains or maintains the performance on varied scoring functions, such as S_{Base}, S_{DeConf}, S_{ODIN}, and S_{Maha}.

The method in this section is orthogonal to all the methods evaluated in this work. For convenience, we will add a * after the name of other methods to indicate a combination, for example, Baseline* and DeConf-C*.

7.4 Experiments

7.4.1 Experimental Settings

Overall procedure: In all experiments, we first train a classifier f_θ on an in-distribution training set, then tune the hyperparameters (*e.g.* the perturbation magnitude ϵ) on an in-

distribution validation set without using its class labels. At testing time, the OoD detection scoring function $S(\boldsymbol{x})$ calculates the scores s from the outputs of f_θ. The scores s is calculated for both in-distribution validation set D_{in}^{val} and out-of-distribution dataset $D_{out} \sim p_{out}$. The scores s are then sent to a performance metric calculation function. The above procedure is the same as related works in this line of research [59, 167, 170, 163, 168, 171], except that we do not use OoD data for tuning the hyperparameters in the scoring function $S(\boldsymbol{x})$.

In-distribution Datasets: We use SVHN [114] and CIFAR-10/100 images with size 32x32 [136] for the classification task. Detecting OoD with CIFAR-100 classifier is generally harder than CIFAR-10 and SVHN, since a larger amount of classes usually involves a wider range of variance, and thus it has a higher tendency to treat random data (*e.g.* Gaussian noise) as in-distribution. For that reason, we use CIFAR-100 in our ablation and robustness study.

Out-of-distribution Datasets: We include all the OoD datasets used in ODIN [59], which are TinyImageNet(crop), TinyImageNet(resize), LSUN(crop), LSUN(resize), iSUN, Uniform random images, and Gaussian random images. We further add SVHN, a colored street numbers image dataset, to serve as a difficult OoD dataset. The selection is inspired by the finding in the line of works that uses a generative model for OoD detection [182, 183, 184]. Those works report that a generative model of CIFAR-10 assigns higher likelihood to SVHN images, indicating a hard case for OoD detection.

Networks and Training Details: We use DenseNet [185], ResNet [139], and WideResNet [186] for the classifier backbone. DenseNet has 100 layers with a growth rate of 12. It is trained with batch size 64 for 300 epochs with weight decay 0.0001. The ResNet and WideResNet-28-10 are trained with batch size 128 for 200 epochs with weight decay 0.0005. In both training, the optimizer is SGD with momentum 0.9, and the learning rate starts with 0.1 and decreases by factor 0.1 at 50% and 75% of the training epochs. Note that we do not apply weight decay for the weights in the $h_i(\boldsymbol{x})$ function of DeConf clas-

sifier since they work as the centroids for classes, and those weights are initialized with He-initialization [187]. In the robustness analysis, the model may be indicated to have an extra regularization. In such case, we additional apply a dropout rate of 0.7 at the inputs for the dividend/divisor structure.

Evaluation Metrics: We use the two most widely adopted metrics in the OoD detection literature. The first one is the area under the receiver operating characteristic curve (AUROC), which plots the true positive rate (TPR) of in-distribution data against the false positive rate (TPR) of OoD data by varying a threshold. Thus it can be regarded as an averaged score. The second one is true negative rate at 95% true positive rate (TNR@TPR95), which simulates an application requirement that the recall of in-distribution data should be 95%. Having a high TNR under a high TPR is much more challenging than having a high AUROC score; thus TNR@TPR95 can discern between high-performing OoD detectors better.

7.4.2 Results and Discussion

OoD benchmark performance: We show an overall comparison for methods that train without OoD data in Table 7.1 with 8 OoD benchmark datasets. The ODIN* and Mahalanobis* are significantly better than the baseline, while DeConf-C* still outperforms them with a significant margin. These results clearly show that learning OoD detection without OoD data is feasible, and the two methods we proposed in Sections 7.3.1 and 7.3.2 combined are very effective for this purpose.

In Table 7.2 we further compare our results with the original ODIN [59] and Mahalanobis [167] methods which are tuned on each OoD dataset. We refer to the results of both original methods reported by [167] since it uses the same backbone network, OoD datasets, and metrics to evaluate OoD detection performance. In the comparison, we find our ODIN* and Mahalanobis* perform worse than the ODIN^{orig} and Mahalanobis^{orig} in a major fraction of the cases. The result is not surprising because the original methods

Table 7.1: Performance of four OoD detection methods. All methods in the table have no access to OoD data during training and validation. ODIN* and Mahalanobis* are modified versions that do not need any OoD data for tuning (see Section 7.2.1). The base network used in the table is DenseNet trained with CIFAR-10/100 (in-distribution data, or ID). All values are percentages averaged over three runs, and the best results are indicated in bold. Note that we only show the most common settings used in literature. The DeConf-C is selected since it shows the best robustness in our analysis, but it is not necessary to perform the best among all DeConf variants. Please see Figure 7.4 and Figure 7.5 for the summary. A more comprehensive version of the table is available in Appendix.

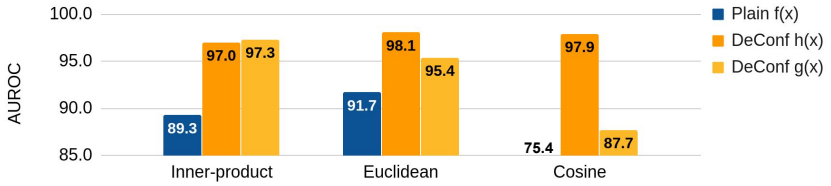
ID	OoD	AUROC	TNR@TPR95
		Baseline / ODIN* / Mahalanobis* / DeConf-C*	
CIFAR-100	Imagenet(c)	79.0 / 90.5 / 92.4 / **97.6**	25.3 / 56.0 / 63.5 / **87.8**
	Imagenet(r)	76.4 / 91.1 / 96.4 / **98.6**	22.3 / 59.4 / 82.0 / **93.3**
	LSUN(c)	78.6 / 89.9 / 81.2 / **95.3**	23.0 / 53.0 / 31.6 / **75.0**
	LSUN(r)	78.2 / 93.0 / 96.6 / **98.7**	23.7 / 64.0 / 82.6 / **93.8**
	iSUN	76.8 / 91.6 / 96.5 / **98.4**	21.5 / 58.4 / 81.2 / **92.5**
	SVHN	78.1 / 85.6 / 89.9 / **95.9**	18.9 / 35.3 / 43.3 / **77.0**
	Uniform	65.0 / 91.4 / **100.** / 99.9	2.95 / 66.1 / **100.** / **100.**
	Gaussian	48.0 / 62.0 / **100.** / 99.9	0.06 / 33.3 / **100.** / **100.**
CIFAR-10	Imagenet(c)	92.1 / 88.2 / 96.3 / **98.7**	50.0 / 47.8 / 81.2 / **93.4**
	Imagenet(r)	91.5 / 90.1 / 98.2 / **99.1**	47.4 / 51.9 / 90.9 / **95.8**
	LSUN(c)	93.0 / 91.3 / 92.2 / **98.3**	51.8 / 63.5 / 64.2 / **91.5**
	LSUN(r)	93.9 / 92.9 / 98.2 / **99.4**	56.3 / 59.2 / 91.7 / **97.6**
	iSUN	93.0 / 92.2 / 98.2 / **99.4**	52.3 / 57.2 / 90.6 / **97.5**
	SVHN	88.1 / 89.6 / 98.0 / **98.8**	40.5 / 48.7 / 90.6 / **94.0**
	Uniform	95.4 / 98.9 / **99.9** / **99.9**	59.9 / 98.1 / **100.** / **100.**
	Gaussian	94.0 / 98.6 / **100.** / 99.9	48.8 / 92.1 / **100.** / **100.**

Table 7.2: OoD detection with OoD data versus without OoD data with CIFAR-10/100 for the in-distribution (ID) data. The values of ODIN^{orig} and Maha^{orig} (abbreviation of Mahalanobis) are copied from the Mahalanobis paper [167] which are tuned with OoD data. The values of ODIN*, Maha*, and DeConf-C* are copied from Table 7.1 of our paper which do not have any access to OoD data. All methods in this table use the same DenseNet for the backbone. Note that the performance with different network backbone may have a mild difference. For example, Maha^{orig} performs slightly better than DeConf-C* with ResNet-34.

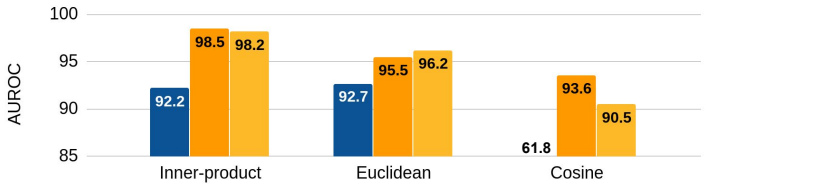
ID	OoD	AUROC	TNR@TPR95
		ODIN^{orig} / Maha^{orig}/ ODIN* / Maha* / DeConf-C*	
C-100	Imagenet(r)	85.2 / 97.4 / 91.1 / 96.4 / **98.6**	42.6 / 86.6 / 59.4 / 82.0 / **93.3**
	LSUN(r)	85.5 / 98.0 / 93.0 / 96.6 / **98.7**	41.2 / 91.4 / 64.0 / 82.6 / **93.8**
	SVHN	93.8 / **97.2** / 85.6 / 89.9 / 95.9	70.6 / **82.5** / 35.3 / 43.3 / 77.0
C-10	Imagenet(r)	98.5 / 98.8 / 90.1 / 98.2 / **99.1**	92.4 / 95.0 / 51.9 / 90.9 / **95.8**
	LSUN(r)	99.2 / 99.3 / 92.9 / 98.2 / **99.4**	96.2 / 97.2 / 59.2 / 91.7 / **97.6**
	SVHN	95.5 / 98.1 / 89.6 / 98.0 / **98.8**	86.2 / 90.8 / 48.7 / 90.6 / **94.0**

gain advantage from using OoD data. However, our DeConf-C* still outperforms the two original methods in many of the cases. The cross-setting comparison further supports the effectiveness of the proposed strategies.

Ablation Study: We study the effect of applying DeConf and our modified input pre-processing (IPP) strategy separately. In Figure 7.4, it shows that both $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$ from all three variants (I, E, C) of the DeConf strategy help OoD detection performance with CIFAR-10 and SVHN classifiers, showing that the concept of DeConf is generally effective. The result of CIFAR-10 is also visualized in Figure 7.7. However, the failure of DeConf-I and $g(x)$ with the CIFAR-100 classifier in Figure 7.5a may indicate these functions have different robustness and scalability, which we will investigate in the next section. One downside of using the DeConf strategy is that the accuracy of the classifier may slightly reduce in the case with CIFAR-100 (A 2% drop compared to a vanilla classifier), summarized in Table 7.3. This could be a natural consequence of having an alternative term, *i.e.* $g(\boldsymbol{x})$, in the model to fit the loss function. This may cause the lack of a high score

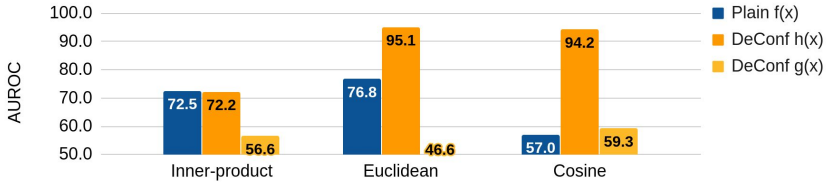


(a) CIFAR-10 classifier

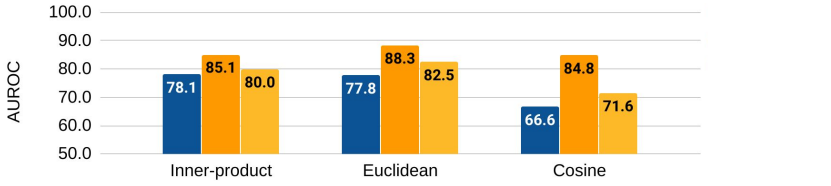


(b) SVHN classifier

Figure 7.4: An ablation study with three variants in our DeConf method (Section 7.3.1). *Plain* means $g(\boldsymbol{x}) = 1$ so that the dividend/divisor structure is turned off. Each bar in the figure is averaged with 24 experiments (8 OoD datasets listed in Table 7.1 with 3 repeats. Note that we use CIFAR-10 as OoD to replace the SVHN in the case of SVHN classifier). The backbone network is Resnet-34. The *plain* setting with inner-product is equivalent to a vanilla Resnet for classification. Overall, both scores from $h(x)$ and $g(x)$ are significant higher than random (AUROC=0.5) and corresponding *plain* baselines. The breakdown results are in Appendix Table A.6



(a) CIFAR-100 classifier



(b) CIFAR-100 classifier with extra regularization (dropout 0.7)

Figure 7.5: An ablation study similar to Figure 7.4. This figure shows the performance of DeConf-I and all $g(\boldsymbol{x})$ are improved by adding extra regularization.

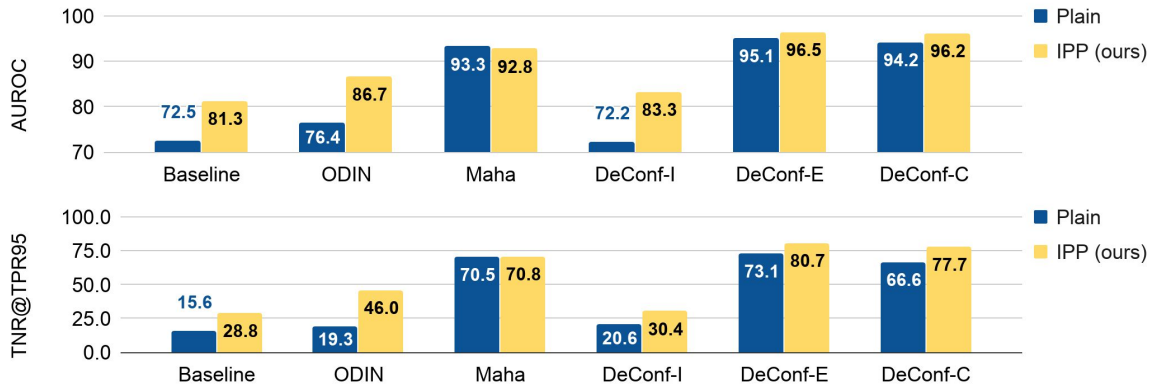
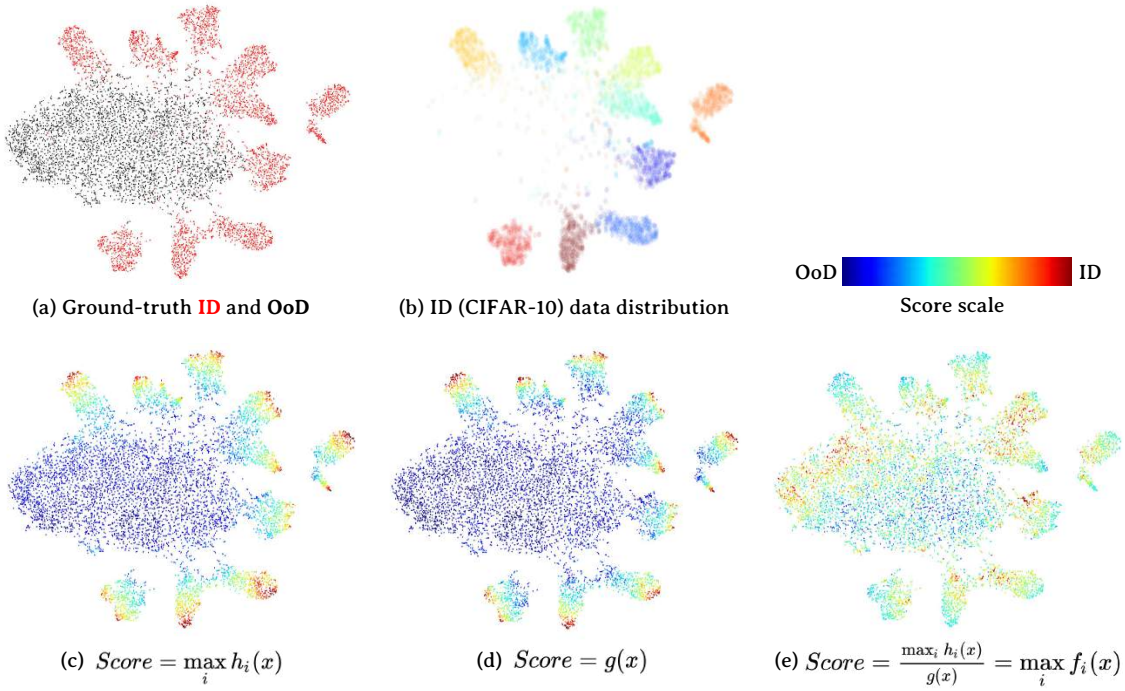


Figure 7.6: The OoD detection performance of our input preprocessing (IPP) strategy, which selects the perturbation magnitude with only in-distribution data. The setting *plain* means the IPP is turned off. The in-distribution data is CIFAR-100. The backbone network is Resnet-34. Each value is averaged with the results on 8 OoD datasets listed in Table 7.1. Each method has its own scoring function $S(\boldsymbol{x})$ (See Section 7.2.1 and 7.3), causing IPP to perform at varied levels of performance gain.

Table 7.3: The summary of classifiers analyzed in this section. Their in-domain classification accuracy is provided in the right four columns. The "+" means that the classifier is trained with extra regularization (dropout rate 0.7). The number in parenthesis is the standard deviation.

Classifier	Image size	#class	Model	Experiment	Baseline	DeConf-I	DeConf-E	DeConf-C
CIFAR10	32x32	10	DenseNet	Table 1,2	95.2±0.1	94.9±0.1	95.0±0.1	95.0±0.1
CIFAR10	32x32	10	ResNet34	Figure 3	95.2±0.1	95.0±0.1	94.9±0.1	95.1±0.1
SVHN	32x32	10	ResNet34	Figure 3	96.9±0.1	96.8±0.1	96.5±0.1	96.7±0.1
CIFAR100	32x32	100	DenseNet	T-1,T-2,F-7	77.0±0.2	75.8±0.4	76.4±0.1	75.9±0.1
CIFAR100	32x32	100	WRN	Figure 7	80.8±0.1	78.3±0.1	78.4±0.1	78.4±0.1
CIFAR100	32x32	100	ResNet50	Figure 7	78.8±0.3	76.4±0.1	76.5±0.3	76.2±0.2
CIFAR100	32x32	100	ResNet34	Figure 4,5,7	78.5±0.2	76.0±0.1	76.2±0.1	75.8±0.2
CIFAR100	32x32	100	ResNet18	Figure 7	77.3±0.1	75.2±0.2	75.8±0.1	75.1±0.1
CIFAR100	32x32	100	ResNet10	Figure 7	75.0±0.1	73.4±0.1	74.2±0.1	73.5±0.1
CIFAR100$^+$	32x32	100	ResNet34	Figure 4	78.2±0.1	77.4±0.3	77.2±0.3	77.2±0.1
DomainNet (Real-A)	180x180 to 640x880	173	ResNet34	Table 3	73.6±0.1	73.0±0.1	73.4±1.5	72.2±0.5



(a) Ground-truth ID and OoD (b) ID (CIFAR-10) data distribution

(c) $Score = \max_i h_i(x)$ (d) $Score = g(x)$ (e) $Score = \frac{\max_i h_i(x)}{g(x)} = \max_i f_i(x)$

Figure 7.7: Visualization of the score distribution. The data are visualized with t-SNE using the features from the penultimate layer of the neural networks. The results are from DeConf-I with ResNet-34. The figure (a) visualizes the ground-truth in-distribution (ID, red, CIFAR-10) and out-of-distribution (OoD, black, Imagenet-resized) data. The colors in (b) represent different classes of CIFAR-10. The scores are obtained from (c) h function, (d) g function, or (e) the logits, and the scores are linearly re-scaled to between zero and one for visualization. The figure presents two phenomena. The first is that the OoD data in (e) have high scores. It is related to the overconfident effect discussed with equation 7.5. The second phenomenon is that high-score data in (c) and (d) are more significantly clustered in each class of CIFAR-10. It shows a tendency that the in-distribution data in high-density regions have higher scores than those in low-density regions (close to OoD data). This phenomenon is related to the discussion at the end of section 7.3.1

.

for $h_i(\boldsymbol{x})$, instead of assigning a lower score for the data away from the high-density region of in-distribution data. We see this effect is reduced and has only a 1% accuracy drop when the extra regularization (dropout rate 0.7) is applied.

In Figure 7.6, the results show that tuning the perturbation magnitude with only in-distribution data is an effective strategy, allowing us to reduce the required supervision for learning. The supervision here means the binary label for in/out-of-distribution.

Robustness Study: This study investigates when the OoD detection method will or will not work. In Figure 7.8, it shows that the number of in-distribution training data can largely affect the performance of the OoD detector. Mahalanobis has the lowest data requirement, but the DeConf methods generally reach a higher performance in the high data regime. In Figure 7.8, we also examine scalability by varying the number of classes in the in-distribution data. In this test, DeConf-E* and DeConf-C* show the best scalability. Overall, DeConf-C* is more robust than the other two DeConf variants. Lastly, Figure 7.9 shows that high performing methods such as DeConf-E*, DeConf-C*, and Mahalanobis* are not sensitive to the type and depth of neural networks. Therefore, *the number of in-distribution samples and classes are the main factors that affect OoD detection performance*.

Enhancing the Robustness: The overfitting issue may be the cause of low OoD detection performance for some of the DeConf variants and $g(x)$. In Figure 7.5b, the OoD detection performance is significantly increased with DeConf-I and all $g(x)$ when extra regularization (dropout rate 0.7) is applied. Figure 7.10 provides further analysis for DeConf-I and its $g(x)$ by varying the number of samples and classes in the training data. The performance with extra regularization is significantly better than the cases without it. Besides, the performance is also very similar between regularized $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$, indicating that overfitting is an important issue. Lastly, we note that the DeConf-E and DeConf-C have a reduced performance with extra regularization in Figure 7.5b. It is an expected outcome since dropout generally harms the distance calculation between centroids and data since part of the feature is masked. The results indicate that the design of (I, E, C) might not be

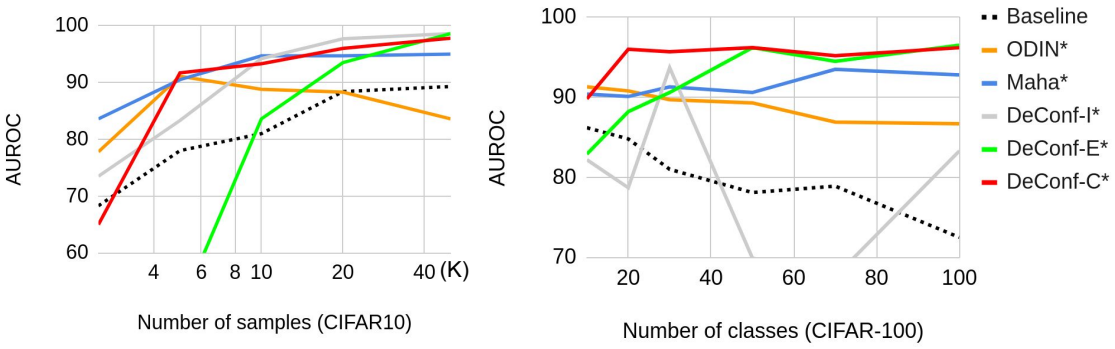


Figure 7.8: Robustness analysis of 6 OoD detection methods. The left figure has classifiers trained on a varied number of samples in CIFAR-10. The right figure has classifiers trained on a varied number of classes in CIFAR-100. Each point in the line is an average of the results on 8 OoD datasets. The backbone network is Resnet-34. Please see Section 7.4.2 for a detailed discussion.

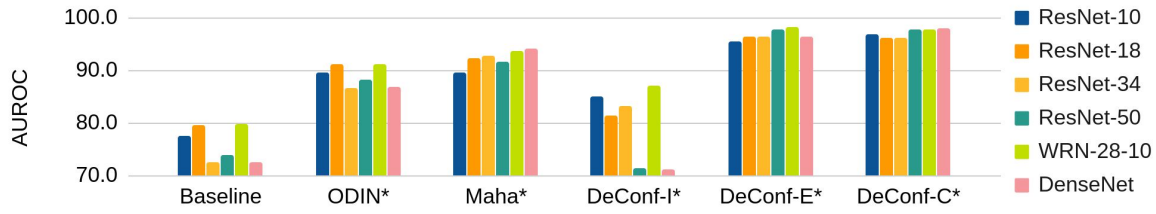


Figure 7.9: Robustness analysis using different neural network backbones. The in-distribution data is CIFAR-100. Each bar is averaged with the results on 8 OoD datasets.

optimal for the problem, leaving room for future work to find a robust pair of $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$ for the OoD detection problem.

7.4.3 Semantic Shift versus Non-semantic Shift

One interesting aspect of out-of-distribution data that has not been explored is the separation of semantic and non-semantic shift. We therefore use a larger scale image dataset, DomainNet [154], to repeat an evaluation similar to Table 7.1. DomainNet has high-resolution images in 345 classes from 6 different domains. There are 4 domains in the dataset with class labels available. They are real, sketch, infograph, and quickdraw, resulting in different types of distribution shifts.

To create subsets with semantic shift, we separate the classes into two splits. Split A has class indices from 0 to 172, while split B has 173 to 344. Our experiment uses real-

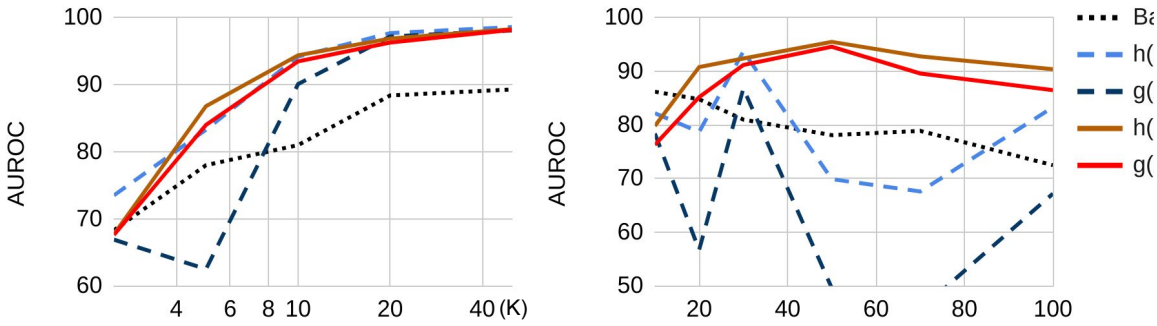


Figure 7.10: Robustness analysis for $h(x)$ and $g(x)$ from DeConf-I. The + sign represents the model trained with extra regularization (dropout rate 0.7).

Table 7.4: Performance of four OoD detection methods using DomainNet. The in-distribution is the real-A subset. Each value is averaged over three runs. The type of distribution shift presents a trend of difficulty to the OoD detection problem: Semantic shift (S) $>$ Non-semantic shift (NS) $>$ Semantic $+$ Non-semantic shift.

OOD	Shift		AUROC	TNR@TPR95
	S	NS	Baseline / ODIN* / Maha* / DeConf-C*	
real-B	✓		**75.1** / 69.9 / 53.6 / 69.8	15.3 / **15.4** / 5.09 / 14.0
sketch-A		✓	75.5 / 80.7 / 59.5 / **84.5**	20.1 / 31.2 / 7.30 / **37.5**
sketch-B	✓	✓	81.8 / 85.7 / 60.4 / **89.1**	25.2 / 36.8 / 7.55 / **44.1**
infograph-A		✓	79.6 / 82.7 / 81.5 / **89.0**	23.5 / 27.8 / 21.6 / **45.4**
infograph-B	✓	✓	82.1 / 85.3 / 80.9 / **90.9**	24.8 / 31.7 / 21.9 / **49.6**
quickdraw-A		✓	78.8 / 96.4 / 67.4 / **96.9**	21.1 / 79.9 / 3.38 / **83.1**
quickdraw-B	✓	✓	80.5 / 96.9 / 66.1 / **97.4**	22.1 / 83.6 / 2.38 / **86.6**
Uniform	✓	✓	54.7 / 75.6 / **99.8** / 99.3	1.65 / 5.37 / **100.** / **100.**
Gaussian	✓	✓	71.3 / 95.5 / **99.9** / 99.4	0.64 / 46.9 / **100.** / **100.**

A for in-distribution and has the other subsets for out-of-distribution. With the definition given in Section 7.2, real-B has a semantic shift from real-A, while sketch-A has a non-semantic shift. Sketch-B therefore has both types of distribution shift. Figure 7.3 illustrates the setup. The classifier learned on real-A uses a Resnet-34 backbone. Its training setting is described in Section 7.4.1 except that the networks are trained for 100 epochs, and the images are center-cropped and resized to 224x224 in this experiment.

The results in Table 7.4 reveal two interesting trends. The first one is that the OoD datasets with both types of distribution shifts are easier to detect, followed by non-semantic

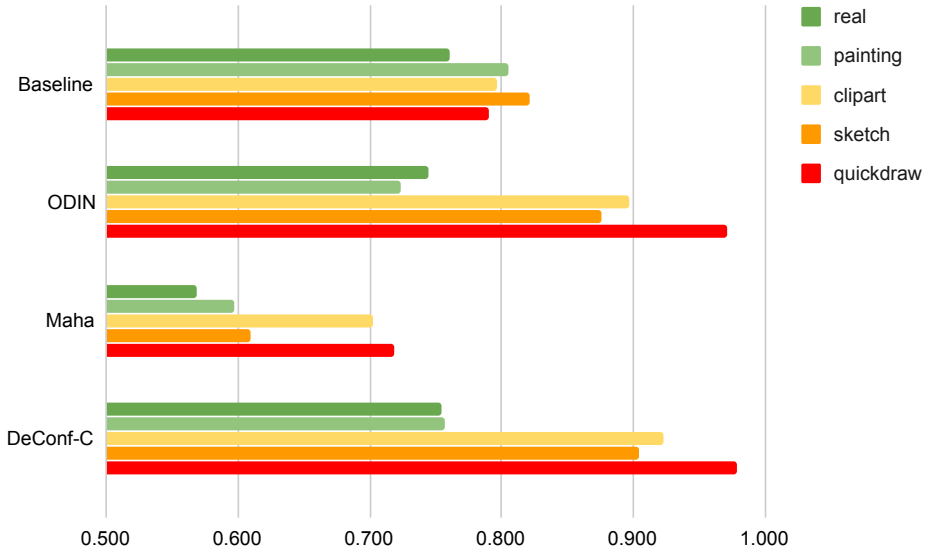
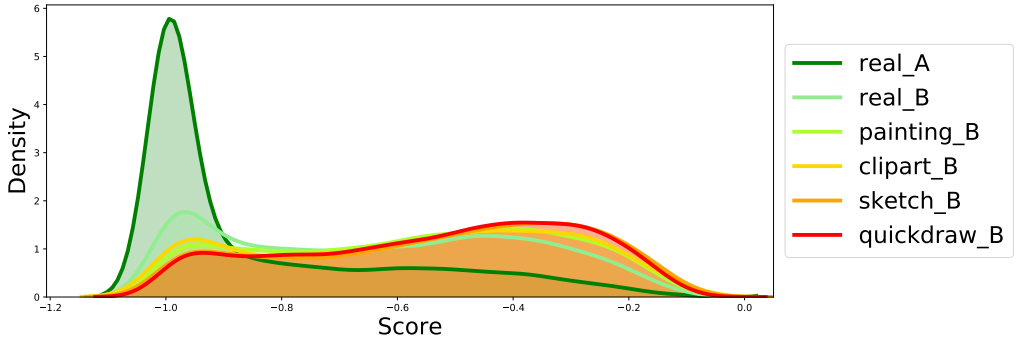


Figure 7.11: The AUROC (higher is better) of OoD detection with the DomainNet dataset. The in-distribution is the first 335 classes from *real* domain, while the OoD data are from five domains. The OoD data are from 10 unseen classes.

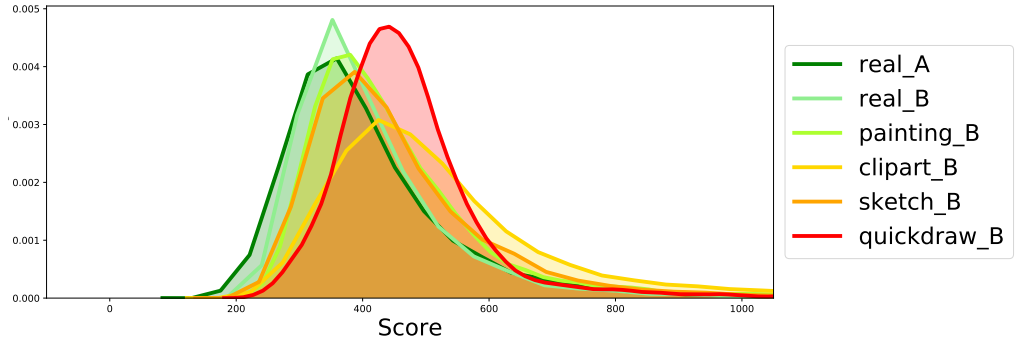
shift. The semantic shift turns out to be the hardest one to detect. The second observation is the failure of Mahalanobis*. In most cases it is even worse than Baseline, except detecting random noise. In contrast, ODIN* has performance gain in most of the cases, but has less gain with random noise. Our DeConf-C* still performs the best, showing that its robustness and scalability is capable of handling a more realistic problem setting, although there is still large room for improvement.

7.4.4 Correlation to Domain Similarity

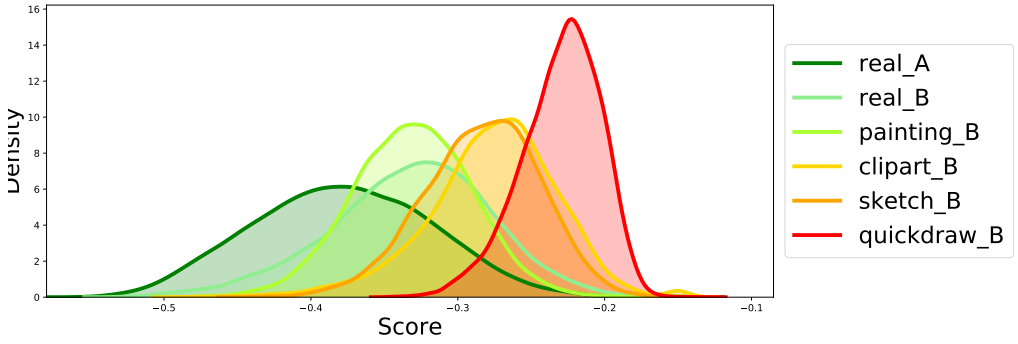
In this section, we revisit the DomainNet experiment setting of Chapter 6, which has 335 classes in the labeled split-A while 10 classes in the unlabeled split-B for category discovery. The goal here is to see if the OoD detection performance (AUROC) is correlated to perceptual domain similarity (approximated by MMD). The results in Figure 7.11 confirms that except the baseline method, other OoD detection methods can detect the perceptually distant domain (*e.g.* quickdraw-B) significantly better than similar domains (*e.g.* real-B and painting-B). Specifically, the Pearson correlation coefficient between DeConf-C's AUROC



(a) Baseline



(b) Mahalanobis*



(c) DeConf-C*

Figure 7.12: Qualitative visualization of the OoD score distribution for each domains in DomainNet. The real-A is the in-distribution data for training the classifier, while the other five domains in split-B are used as OoD data. The results of ODIN* is very similar to DeConf-C with and we only visualize the latter. The better-performing methods (DeConf-C* and ODIN*) have less distributional overlap between in-distribution and OoD data.

and MMD (Figure 6.3) is 0.74, while ODIN-versus-MMD is 0.78, Maha-versus-MMD is 0.69, and Baseline-versus-MMD is 0.08.

Unlike the MMD in Chapter 6, which directly gives a distance between two sets of data, our OoD method gives scores for each data. It makes plotting the distributions be possible. The visualization in Figure 7.12 clearly shows a trend that our method assigns higher OoD scores for the distant domains with less overlap to the data from in-distribution, providing qualitative evidence that our method can estimate the domain similarity better than state-of-the-art OoD methods.

7.4.5 Limitation

In most of the OoD literature as well as ours, the evaluation protocol uses a metric that is threshold-agnostic (*e.g.* AUROC) or has a pre-selected threshold (*e.g.* TNR@TPR95). Such a protocol leaves the problem of *how to select the threshold* open. In order to apply an OoD method to a real-world application, having a method to decide the threshold is necessary. A common way is having some OoD data and in-distribution data in the validation set to make the selection. However, it will again make the whole method prone to selection bias, raising the concern that the selected threshold might not generalize to others. Therefore, developing a principled strategy for threshold selection is an crucial problem for future work.

7.5 Conclusion

In this paper, we propose two strategies, the decomposed confidence and the modified input preprocessing. These two simple modifications to ODIN lead to a significant change in the paradigm, which does not need OoD data for tuning the method. Our comprehensive analysis shows that our strategies are effective and even better in several cases than the methods tuned for each OoD dataset. Our further analysis using a larger scale image dataset shows that the data with only semantic shift is harder to detect, pointing out a challenge for

future works to address. The work of this chapter is published in [12].

CHAPTER 8
APPLICATION: FROM PIXEL-WISE SIMILARITY TO INSTANCE SEGMENTATION

8.1 Introduction

This chapter applies the KCL from Chapter 3 to a real-world application: instance segmentation. The application introduces a major challenge because of the amount of data, which could be millions of pixels in one image. Such a large number is the major barrier to our pairwise-based learning strategy, leading to a squared number of the pairwise combinations. This chapter overcomes this challenge by proposing sampling strategies based on the two-dimensional connectivity constraints, demonstrating the scalability and generalizability of our clustering method. The results are published in [13].

The application, instance segmentation, combines requirements from both semantic segmentation and object detection. It not only needs the pixel-wise *semantic labeling* but also requires *instance labeling* to differentiate each object at a pixel level. Since the semantic labeling can be directly obtained from an existing semantic segmentation approach, most of the instance segmentation methods focus on dealing with the instance labeling problem. This is usually achieved by assigning a unique identifier to all of the pixels belonging to an object instance.

Instance labeling becomes a more challenging task when occlusions occur, or when a vastly varying number of objects in a cluttered scene exist. For example, the top performance (at the time of writing this chapter) on the Cityscapes dataset [188] only reaches 31.8% accuracy [189] (without external training data) in terms of average precision, leaving much room for improvement.

Techniques to solve instance segmentation can be roughly grouped into two categories:

proposal-based methods and *proposal-free* methods. In proposal-based methods [190, 191, 189], a set of object proposals and their classes are first predicted, then foreground-background segmentation of each bounding box is performed. The proposal-free approaches [192, 193, 194, 195, 196] exclude the step of proposal generation. These approaches usually have two stages. The first is to learn an intermediate representation (*e.g.* feature vectors, energy levels, pairwise affinities, breakpoints, or object boundaries) at the pixel level, then in the second stage they group the pixels using a clustering algorithm with the learned representation. Additionally, the proposal-free approaches usually only focus on instance labeling and directly leverage the categorical predictions from semantic segmentation for the semantic labeling.

Our approach belongs to the proposal-free style. We reduce the two-stage paradigm to a single forward pass on a fully convolutional network (FCN) [32]. We achieve this by designing a novel learning objective, which uses pairwise relationships between pixels as the supervision to guide an FCN to learn pixel-wise clustering. The FCN trained with the proposed objective learns to directly assign a cluster index to each pixel, while each pixel cluster is regarded as an object instance. The clustering is done by the forward propagation of the FCN. It turns out the FCN is capable of learning to do pixel-wise clustering and generalize the learned clustering mechanism to unseen images.

The number of cluster indices available in the FCN will limit the number of instances that can be separated by our approach. We provide a strategy to deal with the case of an unlimited number of instances. Inspired by graph coloring theory in how it reuses the indices for coloring a graph, we inject the coloring strategy into our learning objective. Therefore the FCN is trained to assign different indices for the neighboring instances, while reusing the index for the objects that are far away from each other. With the coloring result, each individual instance can be naively recovered by connected components extraction.

We formulate the lane detection problem as an instance labeling problem, and our approach won second place in the lane detection competition of the 2017 CVPR Autonomous

Driving Challenge. The difference of accuracy between ours and first place is insignificant. Considering that the top performer used a large amount of external data for training while we did not, the advantage of our approach becomes even more significant. We are also able to perform the prediction in real-time (~ 55 FPS).

Lane detection is a problem that involves a single category and a limited number of instances; therefore, we extend our evaluation on a multi-category dataset and unlimited instance setting, specifically the Cityscapes dataset. Our approach demonstrates strong performance, achieving 15.1% AP. By comparing to the 9.8% AP of the JGD [197] which shares similar insights in using graph coloring (also called node labeling), our data-driven learning approach has a significant advantage over their search-based algorithm.

In summary, we make several contributions. First, we formulate a novel objective to train an FCN to perform instance labeling. Second, we demonstrate how to combine the graph coloring theorem to augment the learning objective. Third, we empirically show a deep FCN is able to learn to do clustering on image pixels in an end-to-end fashion.

8.2 Related Work

Proposal-based methods: This type of approach usually follows the detect-then-segment paradigm [189, 190, 191, 198, 188, 199, 200, 201, 202] which first detects a bounding box as the object proposal, then segments out the foreground object in the box region. Some variant approaches, for example [203], uses RNNs to generate the proposals instead of using a proposal network. [204] uses the bounding box as a potential in their CRF formulation. Their segmentation performance is restricted by the quality of bounding boxes, which normally favor an instance with a round shape.

Proposal-free methods: Although the approaches in this type share the same two-stage scheme of representation learning then clustering, there is a wide spectrum of ways to achieve it. [192] has a per pixel prediction of breakpoints, then apply a sequential grouping for clustering the pixels. [193] learns an energy level for each pixel and is followed

by watershed clustering. [195] learns a discriminative feature vector for mean shift clustering. [194] uses object boundary prediction with a MultiCut algorithm [205]. [206, 207] learn several hand-picked features then use heuristic or spectral clustering. [197] formulates instance labeling as a node labeling problem and find a feasible solution using a search algorithm. [208] learns position-sensitive score maps, then merge the masks with an assembling module. [196] learns pairwise affinities between pixels followed by a graph merging algorithm for pixel clustering. Our method belongs to this category but is different from above in the way that we do not specify an intermediate representation for learning. We let an FCN [32] learn to perform instance labeling directly.

8.3 Method

In this section, we describe how to formulate the learning objective, and explain how to use a limited amount of indices to label an unlimited amount of instance in an image.

8.3.1 Learning Instance Labeling

The instance labeling task is defined as follows. We have an RGB image as input, and our task is to predict a mask for each instance. This is done by assigning a unique index (instance ID) to all of the pixels in the mask. The index is an integer i, $1 \leq i \leq n$, where n is the number of instances in the scene. One crucial property of the assignment is that it is not unique. Specifically, swapping the index between any two masks will still lead to a valid assignment and equivalent segmentation. This is referred to as the *quotient space property* [209]. The goal of the task is to learn a function f which can assign an index $y_i = f(p_i)$ for a pixel p_i, where $y_i \in \mathbb{Z}$ and i is the index of the pixel in an image. The resultant labeling of all pixels in an image, i.e., $Y = \{y_i\}_{\forall i}$, should fulfill the relationship

R. For any two pixels p_i, p_j, $R(p_i, p_j) \in \{0, 1\}$ is defined as,

$$R(p_i, p_j) = \begin{cases} 1, & \text{if } p_i, p_j \text{ belong to the same instance.} \\ 0, & \text{otherwise.} \end{cases} \tag{8.1}$$

Since the labels in the ground-truth are just one instantiation of the labeling based on the underlying relationship R, we propose to directly use R as the supervision for training. Using R as the learning objective is preferable over using the ground-truth labeling. Since the instance ID of any particular instance is assigned in an ad hoc manner, forcing a particular labeling makes the learning task more difficult because the labeling is not consistent from image to image (*e.g.* a vehicle with similar appearance may be assigned different labels in different images). R is a more precise representation of the actual learning objective. Reconstructing the R from a given labeling is straightforward from equation 8.1.

8.3.1.1 The Learning Objective

We use a fully convolutional neural network (FCN) [32] as f to make pixel-wise prediction. We define the outputs of the FCN as the probability of assigning a pixel to a certain instance index, which is a multinomial distribution. Inspired by Chapter 3, we use the same criteria that if two pixels belong to the same instance, their predicted distributions should be similar, and be dissimilar otherwise. The distance between two distributions could be evaluated by the Kullback-Leibler divergence. Given a pair of pixels p_i and p_j, their corresponding output distributions are denoted as $\mathcal{P}_i = f(p_i) = [t_{i,1}..t_{i,n}]$ and $\mathcal{P}_j = f(p_j) = [t_{j,1}..t_{j,n}]$, where n is the number of indices available for labeling. We therefore rewrite the KCL with the pixel-wise notations of this chapter in below.

$$\begin{aligned} \mathcal{L}(p_i, p_j)^+ &= D_{\mathrm{KL}}(\mathcal{P}_i^\star)||\mathcal{P}_j + D_{\mathrm{KL}}(\mathcal{P}_j^\star||\mathcal{P}_i), \\ \text{where } D_{\mathrm{KL}}(\mathcal{P}_i^\star||\mathcal{P}_j) &= \sum_{k=1}^{n} t_{i,k} log(\frac{t_{i,k}}{t_{j,k}}). \end{aligned} \tag{8.2}$$

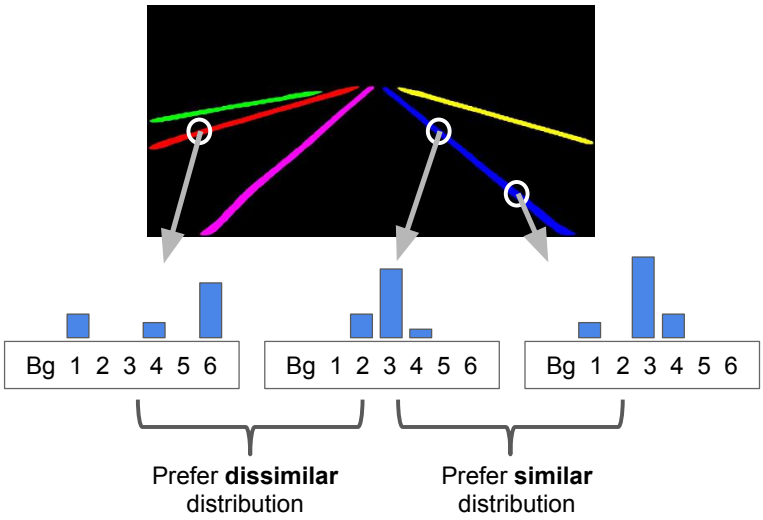


Figure 8.1: The example outputs of lane detection. The colors represent different instance IDs. The outputs for each pixel is a $6+1$ dimensional vector, which represents the probability distribution of this pixel being assigned to a certain ID. Our learning objective (equation 8.4) guides the f to output a similar distribution for the pixels on the same lane line, and vise versa. During testing time, the pixel will be assigned to an ID with highest probability.

$$\begin{aligned} \mathcal{L}(p_i, p_j)^- &= L_h(D_{\text{KL}}(\mathcal{P}_i^\star || \mathcal{P}_j), \sigma) + L_h(D_{\text{KL}}(\mathcal{P}_j^\star || \mathcal{P}_i, \sigma)), \\ &\text{where } L_h(e, \sigma) = max(0, \sigma - e). \end{aligned} \tag{8.3}$$

The margin σ is a hyper-parameter. We use 2, as suggested in Chapter 3, for all our experiments. We then construct a criterion to evaluate how the outputs of f are compatible with R in the form of a contrastive loss:

$$\begin{aligned} \mathcal{L}(p_i, p_j) = R(p_i, p_j)\mathcal{L}(p_i, p_j)^+ \\ + (1 - R(p_i, p_j))\mathcal{L}(p_i, p_j)^-. \end{aligned} \tag{8.4}$$

An example associated with the idea of equation 8.4 is illustrated in figure 8.1. We apply equation 8.4 on top of the outputs of a softmax layer in a standard FCN which was originally designed for semantic segmentation. Therefore the loss function is easy to deploy and combine with other pixel-wise prediction tasks like semantic segmentation and depth estimation.

8.3.1.2 Combining the Sampling Strategy

The objective in equation 8.4 uses pairwise information between pixels. The number of pairs grow quadratically with the number of pixels in an image. Therefore it is not feasible to use all pixels in an image. We adopt a sampling strategy. A fixed total number of pixels (*e.g.* one thousand) is sampled during the training time. Only the pixels in the ground-truth instance masks are picked (see below for how we handle the background class). All instances in an image receive the same number of samples regardless of their size. The pixels in an instance are randomly sampled with uniform distribution. To create the pairs, all possible pairwise relationships between the sampled pixels are enumerated. Therefore one million pairs (including both orders and self-pairs) per image are generated upon which equation 8.4 is applied.

We treat the background as one instance and handle it differently because of its unbalanced nature. Since the background contains the majority of pixels in an image, the sampled points are very sparse. Using a cityscapes [188] image (1024x2048) as example, the density of sampled points on background is roughly 0.005%. This leads to an obvious limitation that the boundary between instance and background is hard to learn. In fact the predicted instances tend to stretch significantly into the background region.

One trivial solution is to increase the number of samples on the background region. We push this notion to the extreme and use all background pixels for training. However, we only consider the unary prediction instead of pairwise relationships. Specifically, we use a binary classification loss for the background, while the background and other instance still share the same output vector which represents the instance index. To achieve that, we reserve the index zero only for the background. Given a $n+1$ dimension predicted outputs $f(p_i) = \mathcal{P}_i = [t_{i,0}..t_{i,n}]$, the summation of non-zero indices $[t_{i,1}..t_{i,n}]$ is the probability of non-background. We formulate the criterion of background classification to be similar to a

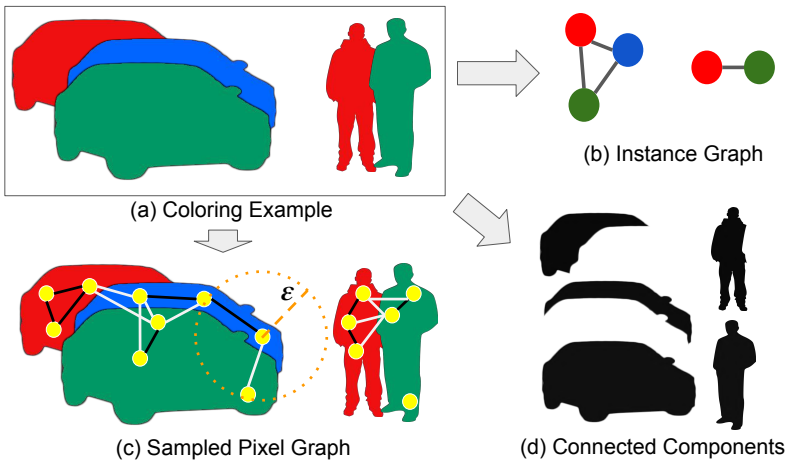


Figure 8.2: The concept of how graph coloring is related to instance ID assignment. For details please see section 8.3.2.

binary cross entropy loss:

$$\mathcal{L}_{bg} = -\frac{1}{N}\sum_{i}^{N}(I_i^{bg}\log t_{i,0} + (1 - I_i^{bg})\log(\sum_{k=1}^{n} t_{i,k})) \tag{8.5}$$

N is the total number of pixels in an image. I_i^{bg} is the indicator function and it returns 1 if pixel i is background. Note that although the value of $\sum_{k=1}^{n} t_{i,k}$ is equal to $1 - t_{i,0}$, the resulting derivative is different and our formulation can encourage the outputs of $[t_{i,1}..t_{i,n}]$ when p_i is not background. Let $T = \{(p_i, p_j)\}_{\forall i,j}$ contain all pairs of sampled pixels, we have the averaged pairwise loss:

$$\mathcal{L}_{pair} = \frac{1}{|T|}\sum_{(p_i,p_j)\in T}\mathcal{L}(p_i, p_j) \tag{8.6}$$

The full formula for instance segmentation is the direct combination of both:

$$\mathcal{L}_{ins} = \mathcal{L}_{pair} + \mathcal{L}_{bg} \tag{8.7}$$

8.3.2 Addressing an Unlimited Number of Instances

The f defined in section 8.3.1 can only represent n instance IDs. Therefore it limits the maximum number of instances that could be detected. Although the fixed amount of instance IDs is sufficient for applications such as lane detection for autonomous driving, it becomes a limitation for datasets like Cityscapes [188] for segmenting an arbitrary number of objects. Although we can deal with the problem by increasing the dimension of the output vector, it will introduce two problems. The first is the distribution of the number of instances in an image usually has a long-tail distribution. Most of the images only contains few instances (ex: 5) and only a small fraction has a large number of objects (ex: 100). In such cases the majority of output nodes will only be trained with a small fraction of training data. Therefore it leads to a poor performance. The second is the efficiency consideration. A high dimension output layer will greatly increase the computation time, since the output has a map size equal to the input image. In this section, we describe a generic approach to labeling an unlimited number of instances with a fixed number of IDs.

Inspired by the graph coloring problem, we reformulate the index assignment task to a graph coloring task. Here we regard the region of instance as a vertex (see figure 8.2 (a)-to-(b)). The distance ϵ between regions decides whether an edge exists or not. The goal of graph coloring is to assign a color to each vertex so that neighboring vertices have different colors. A graph is called k-colorable if we can find an assignment with k or fewer colors. The minumum k of a graph called its *chromatic number*. The k could possibly be much smaller than the number of vertices (the number of instances). For example, if we set the distance threshold ϵ to 1 pixel, there will only be edges between adjacent instances. This case is also called the map coloring problem. According to the four-color theorem [210], we only need four colors to make sure any instances has a color different from its neighbors. The colors mentioned here are equivalent to the set of indices we used to label instance pixels.

A compatible k-colored map means no adjacent instance has the same color. Under

this condition, the individual instance region can be extracted by finding the connected components at the pixel level, i.e., by growing a region which share the same ID. Each connected component (instance segment) will be assigned a unique ID for the final outputs. Figure 8.2 (a)-to-(d) illustrates the process.

8.3.2.1 Learning to do graph coloring

In this section we demonstrate a strategy to train a deep neural network to do graph coloring (also called node labeling). We show that by relaxing the setting of graph coloring, we can deploy the constraints of coloring by only slightly changing the sampling strategy described in section 8.3.1.2.

First, we relax the coloring rules from a constraint that must be satisfied to be a soft guideline. The guideline is "Neighboring instances should have different IDs". It is presented in the learning objective and only used in training stage. Second, we set the distance threshold ϵ to a value larger than 1 pixel (our experiment uses 256). The threshold only applied to the pairs of the randomly sampled pixels in section 8.3.1.2. Compared to the original T that contains all pairs of sampled pixels, the T' only contains the pairs (p_i, p_j) which have spatial distance ($|\overline{p_i p_j}|$) within threshold ϵ:

$$T' = \{(p_i, p_j)\}_{\forall i,j,|\overline{p_i p_j}| \leq \epsilon} \tag{8.8}$$

Therefore the averaged pairwise loss (equation 8.6) becomes:

$$\mathcal{L}_{pair} = \frac{1}{|T'|} \sum_{(p_i,p_j)\in T'} \mathcal{L}(p_i, p_j) \tag{8.9}$$

Figure 8.2(c) demonstrates an example of the sampling. The yellow dots are the sampled pixels. The black edges means its two nodes should have similar predicted label distribution, while the white edges represent the dissimilar pairs. Any two pixels that have distance larger than ϵ are considered to have no edge between them and therefore contribute no loss

at all to the learning objective.

8.3.2.2 Choosing the number of color

The equation 8.6 is a special case of equation 8.9 with $\epsilon = \infty$. With the infinity threshold, there are edges between all instances, so the k has to be equal to the number of object instance in an image. With the decreasing of the ϵ, the chromatic number of the graph is also decreasing. The trend stops at $\epsilon = 1$, which becomes a map coloring problem and has chromatic number 4. Note that it is not necessary to consider the case when ϵ is smaller than one pixel. In that case all instances are independent, i.e., no edge between any vertices, therefore one color is sufficient to color the graph. However, individual instance pixels can't be extracted with the resulting coloring.

Since we transform the coloring constraints to a soft learning objective, the choice of n has no hard requirement. Based on the arguments in above paragraph, setting n to any number larger or equal to four could be sufficient, and it is also dependent on the setting of ϵ. We determine the two parameters empirically.

8.3.3 Combinations of Strategies

We consider two factors in instance segmentation applications, which are limited/unlimited number of instances and single/multiple categories.

For applications with a limited number of instances, applying the approach in section 8.3.1 is sufficient. One example is lane detection for autonomous vehicles, which usually has a bounded number of visible lanes in the camera view. The benefit of applying section 8.3.1 standalone is that it is a fully end-to-end solution that can be accomplished in a standard FCN. No post-processing is required. In contrast, when the number of instances is unlimited, the approaches in section 8.3.1 and 8.3.2 are both applied. Connected component extraction is then needed as a post-processing to generate the final predictions.

For the case of multiple categories, we note that our instance ID assignment approach

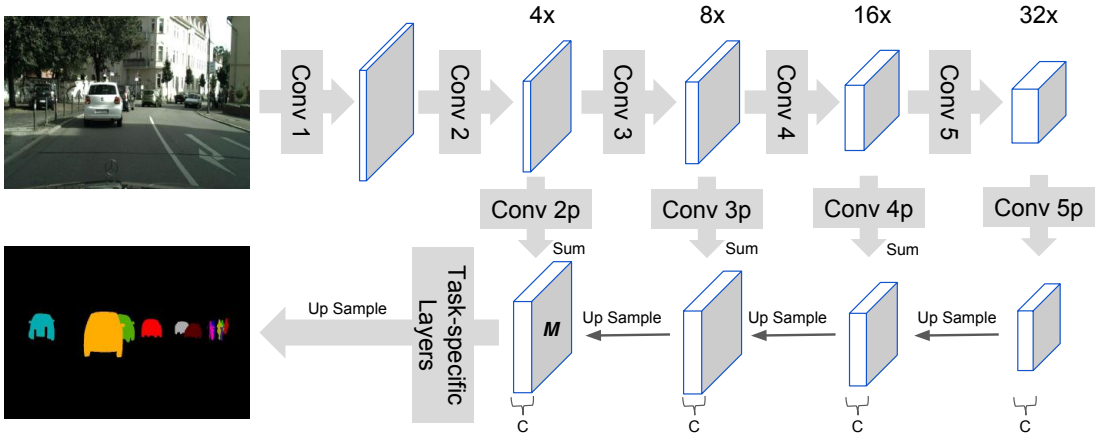


Figure 8.3: The network architecture used in this work.

is category agnostic. Therefore it needs external information to help assigning the class to each instance. For each instance mask, we average the predicted semantic segmentation probability in the masked region and find the dominant category. The intersection between our instance mask and the dominant category mask of semantic segmentation is used as the final instance output. Since we use FCN for the f, it is straightforward to add an output branch to predict the semantic segmentation while sharing most of the layers except the final layer.

8.3.4 Network Architecture

This section describes the network architecture used for f. Figure 8.3 illustrates the diagram. This style of FCN is widely used in pixel-wise prediction and was referred as FPN[211]. The major benefit of using FPN is its configurable dimension for the pixel-wise feature map. In our implementation, the layers Conv-1 to Conv-5 have the weights initialized by pre-trained ResNet [112]. The Conv-2p to Conv-5p (called Conv-Xp for abbreviation) have kernel size 3x3 and are followed by batch normalization [96] and ReLU. The Conv-Xp layers have the outputs of channel dimension c, which is configurable. The outputs of Conv-Xp layers are up-sampled and have element-wise summation with the outputs from lower layers. The resulting feature map M has c feature channels and is four

times smaller than the input image. Furthermore, since we use element-wise summation to combine the features from different convolution blocks, the Conv-Xp work like learning the residual representation for constructing the M.

The task-specific layers are added on top of M. For the instance ID assignment task, we use two convolution layers. The first one has 3x3 kernel and c output channels, followed by batch normalization and ReLU. The second one has 1x1 kernel with $n + 1$ dimension outputs, which maps to n instance IDs and one background ID. Other pixel-wise prediction tasks can also be added here to construct a multi-head structure for multi-task learning; for example, semantic segmentation, boundary detection, depth estimation, and object center prediction. Those tasks can reuse the same two-layer structure by only changing the number of final outputs to fit their target number of categories. In our evaluation on Cityscapes dataset, we add semantic segmentation to help assign the category of each object instance.

8.4 Experiments

In this section, we evaluate the performance of our proposed method on two vastly different datasets. The first one is a lane detection dataset and the second is the benchmark Cityscapes dataset. Our submission to the lane detection competition won the 2nd place, evaluated for both performance and speed, while on the Cityscapes instance segmentation results our entry is among the top 10 of all entries, and among the top four proposal-free entries.

8.4.1 Lane Detection

The Tusimple dataset [212] contains 3626 video clips of highway driving scenes in the training set. One image from each video clip is annotated with ground-truth lane lines. The number of lane lines vary from 3 to 5. The lane lines are labeled in arbitrary order. The task is to predict all individual lane lines for the test set of 2782 images. We can consider the lane lines as a thin and long region on the image. Therefore it becomes a single category

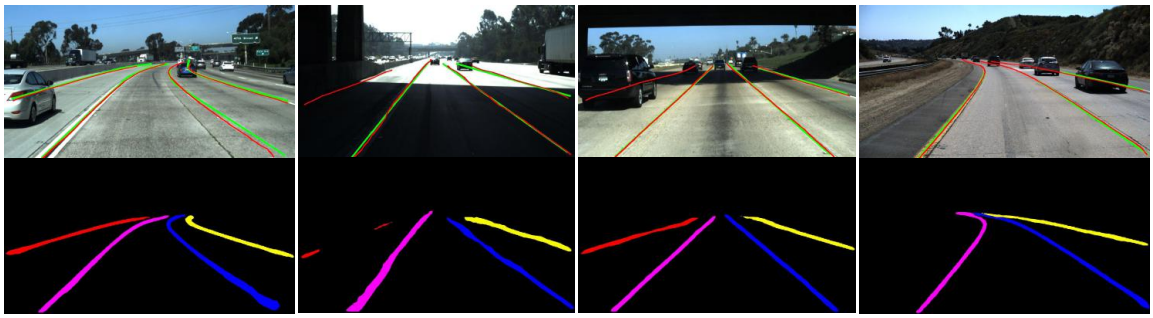


Figure 8.4: The visualization of the lane detection on Tusimple dataset (our validation split). The red lines in top row are our predictions, while the green lines are the ground-truth. The second row shows the raw outputs from our network. The colors represent the assigned IDs.

multi-instance segmentation task.

The evaluation metric used by the challenge is a recall with penalty on extra detections. The recall score is calculated as

$$\text{score} = \frac{\text{number of matched lane points}}{\text{number of ground-truth lane points}} \tag{8.10}$$

The detected lane lines are not densely evaluated per pixel, but rather sampled with horizontal lines spaced every 10 pixels. The sampled points are then compared with sampled points in the ground-truth. If their distance is below 20 pixels, it is considered a matched point. The score for each lane line is computed as above and then averaged to give the final score for an image. Since recall is biased toward methods with many detections, the final score also penalizes extra detections beyond $N + 2$, where N is the number of lines in ground-truth. Submissions must also achieve a minimum speed of 5 FPS on a single GPU to be accepted.

The key challenge of this competition is to correctly predict the number of lines and their exact position. We formulate it as an instance segmentation problem by drawing the lane lines with 10 pixel-width. In that way we obtain a thin and long mask for each lane lines. Since there are at most 6 lane lines in the dataset, it is a problem with a limited

Table 8.1: Results of the top five performers of the 2017 CVPR lane detection challenge [213]. FP: False Positive. FN: False Negative. Ext. data: Use external labeled training data.

User ID/Method	Accuracy	FP%	FN%	Ext. data
XingangPan[214]	96.53%	6.17	1.80	yes
Ours	**96.50%**	**8.51**	**2.69**	**no**
DavyNeven[195]	96.40%	23.65	2.76	N/A
xxxxcvcxxxx	96.14%	20.33	3.87	N/A
TF Placeholder	95.96%	6.54	4.23	N/A

amount of instances. We designed the network output to be a 7-D vector at each pixel location, representing the probability of the pixel belonging to a particular index, including background. The loss function used here is only the equation 8.7.

8.4.1.1 Experiment Setting

The network has a backbone of ResNet-18 [112] with configuration $c = 32$. To train the network, we split the training images into 80% for training and 20% for validation. We applied standard data augmentation (*e.g.* horizontal flipping and color jittering) on the training images. We sampled 100 pixels from each line to compute the equation 8.7. The stochastic gradient descent is used to optimize the proposed learning objective with initial learning rate 0.01, which decays per 20 epochs with factor 0.1 until 50 epochs. During testing, the outputs of the network have cluster indices assigned to all pixels, while each cluster index corresponds to a line (see figure 8.4). For benchmarking purpose, the mean x-coordinate of each line at specific hight is calculated to produce the exact submission format.

8.4.1.2 Results and Discussion

Table 8.1 shows the top 5 performers among 14 teams of the lane detection challenge. The accuracy is defined in equation 8.10. False-positive and false-negatives are also listed

Table 8.2: The AP results on Cityscapes test set. Only the proposal-free approaches are listed.

Method	AP	AP50%	AP100m	AP50m
SGN [192]	25.0	44.9	38.9	44.5
DWT [193]	19.4	35.3	31.4	36.8
DL [195]	17.5	35.9	27.8	31.0
InstanceCut [194]	13.0	27.9	22.1	26.1
JGD [197]	9.8	23.2	16.8	20.3
Uhrig et al. [197]	8.9	21.1	15.3	16.7
ours	15.1	30.8	24.2	25.8

for reference. Our method won the second position and is the top performer without using labeled external training data. The top performer XinganPan uses the approach that requires a specifically designed layer, *e.g.* the SCNN [214], and needs the lanes labeled in certain order, *e.g.* from left to right, so it can uses a standard cross entropy loss to classify the lines. In contrast, our approach only uses standard convolution layers and the lanes can be presented in random order. Therefore our method can largely simplify the labeling effort for constructing the training data. The third performer DavyNeven also utilizes an instance segmentation strategy [195]. Its learning objective learns the embedding of pixel feature vector and therefore needs extra post-processing to cluster the pixels for discovering the lines. It can be non-trivial to make the hyper-parameters of the predefined clustering algorithm perform well. And it is hard to decide the number of road lanes. In contrast, our network performs clustering in an end-to-end fashion and can predict the active clusters with very few false positives, therefore it shows a significant advantage over DavyNaven in terms of FP%.

8.4.2 Cityscapes Instance Segmentation

The Cityscapes dataset [188] has high quality instance segmentation annotation for 8 different object classes. It is a common benchmark for comparing instance segmentation

Table 8.3: AP results on Cityscapes validation set.

Method	AP	person	rider	car	truck	bus	train	motorcycle	bicycle
DWT	21.2	15.5	13.8	33.1	27.1	45.2	14.5	11.9	8.8
DWT + Oracle Ranking	27.6	20.6	18.7	40.1	31.5	50.6	28.3	17.4	13.4
Ours	16.0	14.3	12.7	25.5	13.5	27.0	13.6	10.7	11.0
Ours + PSPnet-Seg	18.6	15.2	15.6	26.5	15.0	35.3	19.4	10.5	10.8
Ours + GT-Seg	28.4	25.0	35.8	31.8	22.3	42.1	27.1	22.8	20.2
Ours + Oracle Ranking	23.0	19.5	17.6	33.7	20.3	37.1	24.8	15.9	15.4
Ours + PSPnet-Seg + Oracle Ranking	25.2	20.3	20.5	33.9	21.3	46.4	27.8	16.4	15.0
Ours + GT-Seg + Oracle Ranking	38.4	33.2	46.3	40.6	30.2	54.4	38.3	36.3	27.8

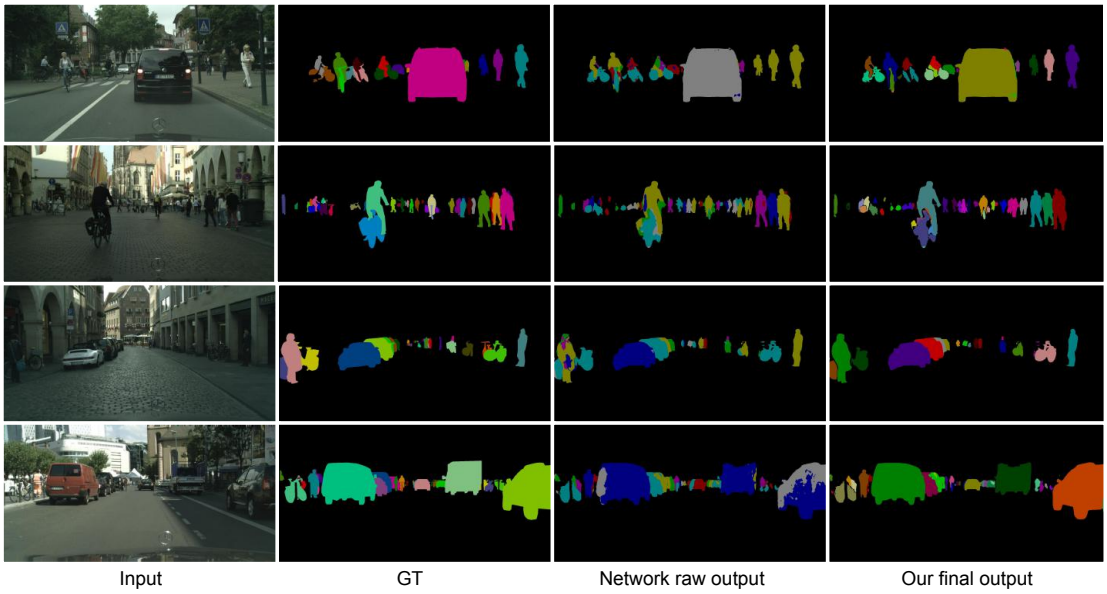


Figure 8.5: Sample outputs of our model on Cityscapes validation set. The colors represent different instance IDs. GT is the ground-truth. Network outputs has eight colors. The rightmost column is the final outputs after connected component extraction and merging.

performance. Cityscapes is a more challenging dataset than the lane detection dataset in three ways. First, lane lines are relatively well structured while objects in Cityscapes have arbitrary shape, scale, and location. Secondly, the number of objects in Cityscapes is larger and unbounded. Lastly, Cityscapes contains multiple categories. Therefore it is a target to demonstrate the generalizability of our approach.

8.4.2.1 Experiment Setting

We use the official splits of training, validation, and testing set, which have 2975, 500, and 1525 images, respectively. For evaluation, we also use the official scoring, which calculates the average precision (AP) with various intersection-over-union (IoU) thresholds, i.e., 50% to 95% with step size 5%, between predicted instances and ground truth instances. Additionally, we report the AP at 50% overlap, AP of objects closer than 50m, and AP of objects closer than 100m.

The network used here has a backbone of pre-trained ResNet-101. The feature dimension c is set to 512. To enlarge the field of view, we add the pyramid pooling module [215] after Conv-5. Our pyramid pooling has the same four pooling scales as [215], but we do up-sampling and element-wise sum with the 32x feature map in figure 8.3, instead of projection and concatenation in the original design. For the purpose of obtaining the category of instance and post-processing, we add two extra task-specific modules on top of the feature map M. One is for semantic segmentation and the other is for predicting the object center. The later module has 2-D output which corresponds to the vector pointing to object center from a specific pixel. Its usage is described in the next section.

For training the network, we sample 50 pixels from each object. The loss $\mathcal{L}_{pair}$ has the form of equation 8.9, while the ϵ is set to 256 pixels and the n is set to 8, which are tuned with the validation set. We use the cross entropy loss for the semantic segmentation and use the smooth L1 loss for the regression of object center prediction. The weights for the instance ID assignment, semantic segmentation, and object center prediction are 1, 0.1,

0.01, respectively. We use stochastic gradient descent to optimize the three losses jointly with learning rate 0.01, which decays per 30 epochs with factor 0.1. The training proceeds for 90 epochs.

8.4.2.2 Post-processing

Each instance obtains a category from the prediction of semantic segmentation. The calculation is described in section 8.3.3. Since we apply the graph coloring strategy for the unlimited number of instance, the connected component extraction has to be applied. Therefore the occluded object might be separated into multiple masks after the step. Here we use the predicted object center to reunion those segments. The average predictive object center is first obtained for each segment, then two segments are merged if their average center are within 20 pixels, which is tuned with the validation set. The merge operation not only helps the occlusion case, but also the situation that an object is separated into several segments due to its large size.

To calculate the AP, it requires a confidence score for each instance. Similar to SGN [192] and DWT [193], we assign confidence value 1 to all our predictions, except for the instances which have size smaller than a threshold (*e.g.* 1500 pixel). In the later case, its confidence score is its region size (in pixel) divided by the threshold.

8.4.2.3 Results and Discussion

Table 8.2 summarizes the results on test set. We ranked forth among the proposal-free approaches. Our method (15.1%) has significant advantage over the JGD (9.8%) [197] which also leverages the graph labeling concept.

We analyze the effect of semantic segmentation quality in table 8.3. Since the semantic segmentation is used to decide the category of each pixel, it plays a substantial rule to affect the AP score. Three semantic segmentations are compared, which are the semantic segmentation outputs from our network (row 3), the prediction from PSPnet [215] (row 4),

and the ground-truth (row 5). The results show a clear trend that the AP increased (from 16.0% to 28.4%) as the semantic segmentation enhanced. This is despite the fact that the same set of our network outputs is used in all the evaluations from row 3 to row 8.

We also evaluate the effect of the confidence score. The oracle ranking is used in table 8.3 row 6 to row 8. When it is combined with ground-truth semantic segmentation, the AP of our instance masks could reach 38.4%. It also explains why the qualitative result in figure 8.5 are visually appealing but it gets fair AP score in the benchmark. The limitation of using AP to evaluate instance segmentation is also discussed in [193].

Besides the effect of semantic segmentation and confidence ranking, another dominant failure mode is that neighboring segments are assigned with the same ID. An example is the third row in figure 8.5 which merges adjacent cars. Another defect is the over-segmentation of large object, for example, the cars in the last row, but such a problem is usually mitigated by the merging step. We leave possible enhancement for future work.

8.5 Conclusion

We proposed a novel objective to train a network to perform a clustering-based instance labeling. By adjusting the sampling method, we are able to inject the graph coloring strategy into the learning objective. The strong performance on two vastly different datasets demonstrates the generalizability and applicability of proposed learning strategy.The work is published in [13]. One possible futrue work is replacing the KCL by MCL.

CHAPTER 9
CONCLUSION

This thesis demonstrates that in image clustering and classification problems, pairwise similarity can be the primary supervision for learning. We provide evidence that pairwise similarities enable deep neural networks to perform clustering without a given number of clusters (Chapter 3), and learn a classifier without class labels (Chapter 5). In the latter case, we show that the resulting classification accuracy is comparable to the vanilla setting of learning with class labels, although the mapping between classifier outputs and classes need to be found with a few class labels. These results empower several strategies toward reducing the labeling effort, discussed in Chapter 1. The first is converting a class labeling task to a binary decision problem for human labelers, illustrated in Figure 1.1. The second uses natural cues such as spatial or temporal relationships to create pairwise similarities for free. The third one transfers a learned similarity measure from labeled data to unseen data, which may or may contain known classes, allowing us to perform category discovery.

The thesis focuses on the third strategy, exploring all three aspects of transfer learning. Given the learned pairwise similarity function for *what to transfer*, we propose methods for unsupervised cross-task and cross-domain transfer learning in answering *how to transfer* (Chapter 4). The results show that novel classes are discovered with a classification accuracy significantly better than previous methods. It indicates an advantage of our method in mimicking human capability, which identifies semantically coherent clusters by leveraging a-priori knowledge of how categories are defined. Although this transfer scheme requires a large number of classes in the source dataset for learning a generic similarity measure, it doesn't limit our core methods, KCL and MCL, from using the first or second strategies for collecting the pairwise similarities.

For the question of *when to transfer*, we confirmed the correlation between domain sim-

ilarity and cross-task transfer learning performance (Chapter 6), providing new strategies for detecting data from a different distribution (Chapter 7). Our methods achieve state-of-the-art performance on out-of-distribution detection benchmarks, although there is still a room for improvements with a larger scale dataset.

Lastly, we confirm the scalability and generalizability of our constrained clustering method on instance segmentation problems (Chapter 8). The pairwise similarity is utilized at the pixel-level, which has millions of samples, demonstrating the applicability of our methods in a real-world setting.

9.1 Future Research Directions

This thesis focuses on how to use pairwise similarity, but not how to collect it or improve its quality for learning. Therefore, answering *how to learn pairwise similarity* more effectively will lead to works complementary to this thesis. For example, the similarity prediction function is a fixed module in our demonstration. If this function adapts to the target domain, the accuracy of pairwise similarity prediction could be improved. As a result, the downstream task, constrained clustering, will be improved as well. Another example is using a more complicated model to learn the pairwise similarity function. This direction includes finding better deep neural networks which capture the similarity better between a pair of data, or using alternative learning objectives such as triplet loss [216], quadruplet loss [217], or lifted structured feature embedding [218] for the similarity prediction.

Another direction is to explore the out-of-distribution detection problem further. Our method, Decomposed Confidence, is a new perspective for the problem and can be implemented in different ways. For example, the decomposed confidence effect might be easier to obtain when a proper regularization is added, which needs more investigation. In addition, another method, the modified input preprocessing, could be further generalized by having a generative model for it instead of analytically calculating the amount of input perturbation based on gradients.

The above two points are closely related when viewed from a higher level of abstraction. The first point of learning a better pairwise similarity function is a problem of how to learn the knowledge so that it can generalize to different types of distribution shifts. The second point of improving OoD detection is a problem of detecting when the learned similarity function can not generalize. The solutions to these two problems are adversarial to each other. Making an interaction between the two problems might lead to a new perspective for model generalizability.

Lastly, continual learning [219] is getting more attention, which aims to enable an autonomous agent to learn incrementally without forgetting what it has already learned. Transfer learning is a crucial technique for incremental learning. The agent also needs to know what it does not know (out-of-distribution detection), so that it can initiate a knowledge discovery process (such as our category discovery method) or actively ask for human supervision to expand its capability. The methods proposed in this thesis can be used as building blocks for continual learning problems, enriching the set of strategies that one can leverage. Additionally, we have made our work on benchmarking varied continual learning scenarios [220] public, setting up baselines for the problem. We hope such work together with this thesis inspires more works toward pushing the boundaries of the field.

Appendices

APPENDIX A

ADDITIONAL EXPERIMENT RESULTS

Table A.1: A breakdown of the results for each alphabet in $Omniglot_{eval}$. The unsupervised cross task transfer experiment is described in section 4.5.1. This table shows the clustering accuracy with $K = 100$ to simulate the situation of unknown number of clusters.

Alphabet	K-means	LPNMF	LSC	ITML	SKMS	SKKm	SKLR	CSP	MPCK-means	CCN
Angelic	24%	26%	27%	48%	78%	40%	38%	72%	50%	**82%**
Atemayar_Qelisayer	19%	14%	17%	51%	61%	51%	44%	61%	50%	**82%**
Atlantean	19%	15%	19%	57%	72%	55%	51%	**77%**	47%	72%
Aurek_Besh	23%	18%	21%	45%	49%	44%	45%	76%	71%	**90%**
Avesta	22%	18%	19%	47%	29%	39%	38%	65%	52%	**76%**
Ge_ez	19%	17%	17%	56%	58%	47%	42%	72%	55%	**82%**
Glagolitic	22%	19%	21%	42%	38%	58%	62%	66%	61%	**85%**
Gurmukhi	15%	13%	15%	44%	26%	54%	48%	60%	56%	**80%**
Kannada	18%	14%	16%	48%	37%	46%	53%	60%	48%	**62%**
Keble	20%	14%	18%	44%	60%	46%	44%	75%	68%	**90%**
Malayalam	18%	15%	16%	36%	24%	49%	52%	50%	54%	**72%**
Manipuri	17%	15%	16%	49%	40%	53%	54%	66%	62%	**85%**
Mongolian	18%	18%	19%	50%	40%	37%	48%	75%	57%	**86%**
Old_Church_Slavonic_Cyrillic	19%	16%	19%	42%	41%	52%	67%	71%	67%	**94%**
Oriya	15%	13%	14%	46%	40%	54%	45%	57%	52%	**67%**
Sylheti	14%	13%	14%	49%	32%	35%	37%	59%	43%	**64%**
Syriac_Serto	20%	18%	19%	55%	70%	42%	35%	70%	47%	**73%**
Tengwar	18%	18%	18%	51%	44%	49%	40%	61%	44%	**66%**
Tibetan	18%	14%	16%	44%	31%	47%	54%	59%	55%	**84%**
ULOG	20%	18%	18%	41%	41%	40%	41%	56%	38%	**70%**
Average	19%	16%	18%	47%	46%	47%	47%	65%	54%	**78%**

Table A.2: The performance of unsupervised transfer across domains on Office-31 dataset. The backbone networks in the comparison have different numbers of convolutional layers. AlexNet has 5 layers and the ResNets have 18∼50 layers. SO is the abbreviation for source-only, which simply trains on S' and directly applies the classifier on T. The first two rows are directly copied from [43]. The features learned with deeper networks generalize better across domains.

	A → W	D → W	W → D	A → D	D → A	W → A	Avg	Gain
AlexNet SO	61.6	95.4	99.0	63.8	51.1	49.8	70.1	-
AlexNet (JAN-A)	**75.2**	96.6	**99.6**	72.8	57.5	56.3	76.3	+6.2
Resnet-18 SO	66.8	92.8	96.8	67.1	51.4	53.0	71.3	+1.2
Resnet-34 SO	73.1	96.4	98.8	73.8	**63.2**	**62.1**	77.9	+7.8
Resnet-50 SO	74.3	**96.8**	98.8	**79.5**	61.2	60.6	**78.5**	**+8.4**

Table A.3: Estimates for the number of characters across the 20 datasets in $Omniglot_{eval}$ when C is unknown. The bold number means the prediction has error smaller or equal to 3. The number of dominant clusters is defined by $NDC = \sum_{i=1}^{K} [C_i >= E[C_i]]$, where $[\cdot]$ is an Iverson Bracket and C_i is the size of cluster i. For example, $E[C_i]$ will be 10 if the alphabet has 1000 images and $K = 100$. The $ADif$ represents average difference [134].

Alphabet	#class	SKMS	KCL	MCL
Angelic	20	16	26	**22**
Atemayar Q.	26	17	34	**26**
Atlantean	26	21	41	**25**
Aurek_Besh	26	14	**28**	22
Avesta	26	8	32	**23**
Ge_ez	26	18	32	**25**
Glagolitic	45	18	**45**	36
Gurmukhi	45	12	**43**	31
Kannada	41	19	**44**	30
Keble	26	16	**28**	**23**
Malayalam	47	12	**47**	35
Manipuri	40	17	**41**	33
Mongolian	30	**28**	36	**29**
Old Church S.	45	23	**45**	38
Oriya	46	22	**49**	32
Sylheti	28	11	50	**30**
Syriac_Serto	23	19	38	**24**
Tengwar	25	12	41	**26**
Tibetan	42	15	**42**	34
ULOG	26	15	40	**27**
$ADif$		16.3	6.35	**5.1**

Table A.4: Performance of six OOD detection methods on 8 benchmark datasets. This is a full version of Table 7.1, which uses DenseNet for the backbone networks. The value in parentheses is the standard deviation.

ID	OOD	AUROC
		Baseline / ODIN* / Maha* / DeConf-I* / DeConf-E* / DeConf-C*
CIFAR-100	Imagenet(c)	79.0(2.2) /90.5(1.1) /92.4(0.3) /84.4(2.3) /95.1(0.5) /97.6(0.2)
	Imagenet(r)	76.4(3.2) /91.1(1.3) /96.4(0.2) /81.2(3.6) /97.4(0.3) /98.6(0.2)
	LSUN(c)	78.6(1.1) /89.9(0.5) /81.2(0.6) /91.7(0.3) /90.1(0.3) /95.3(0.4)
	LSUN(r)	78.2(2.4) /93.0(0.8) /96.6(0.2) /84.1(2.1) /97.8(0.2) /98.7(0.0)
	iSUN	76.8(2.7) /91.6(1.1) /96.5(0.2) /82.1(2.9) /97.4(0.2) /98.4(0.0)
	SVHN	78.1(3.5) /85.6(0.0) /89.9(0.2) /89.7(0.4) /94.0(0.6) /95.9(0.7)
	Uniform	65.0(22.) /91.4(10.) /100.(0.0) /48.5(16.) /99.9(0.0) /99.9(0.0)
	Gaussian	48.0(28.) /62.0(38.) /100.(0.0) /6.79(4.9) /99.9(0.0) /99.9(0.0)
CIFAR-10	Imagenet(c)	92.1(1.0) /88.2(4.2) /96.3(0.1) /98.2(0.0) /98.0(0.2) /98.7(0.1)
	Imagenet(r)	91.5(1.4) /90.1(4.1) /98.2(0.0) /98.4(0.0) /98.2(0.2) /99.1(0.1)
	LSUN(c)	93.0(0.5) /91.3(2.0) /92.2(0.4) /98.4(0.0) /98.6(0.2) /98.3(0.2)
	LSUN(r)	93.9(0.4) /92.9(2.9) /98.2(0.0) /98.6(0.0) /98.8(0.0) /99.4(0.1)
	iSUN	93.0(0.7) /92.2(3.4) /98.2(0.0) /98.6(0.0) /98.8(0.0) /99.4(0.0)
	SVHN	88.1(4.8) /89.6(0.3) /98.0(0.3) /98.2(0.2) /98.4(0.6) /98.8(0.1)
	Uniform	95.4(0.7) /98.9(0.7) /99.9(0.0) /99.2(0.5) /99.9(0.0) /99.9(0.0)
	Gaussian	94.0(2.9) /98.6(1.7) /100.(0.0) /99.1(0.3) /99.9(0.0) /99.9(0.0)
ID	OOD	TNR@TPR95
		Baseline / ODIN* / Maha* / DeConf-I* / DeConf-E* / DeConf-C*
CIFAR-100	Imagenet(c)	25.3(2.8) /56.0(3.1) /63.5(2.1) /31.0(3.4) /74.6(2.8) /87.8(1.7)
	Imagenet(r)	22.3(3.1) /59.4(3.7) /82.0(1.6) /21.4(4.0) /87.6(1.7) /93.3(1.2)
	LSUN(c)	23.0(2.2) /53.0(1.0) /31.6(1.3) /59.6(1.9) /51.0(1.0) /75.0(1.9)
	LSUN(r)	23.7(2.5) /64.0(3.0) /82.6(1.8) /21.1(3.3) /89.8(1.5) /93.8(0.3)
	iSUN	21.5(2.8) /58.4(4.1) /81.2(1.4) /17.6(3.3) /87.3(1.2) /92.5(0.2)
	SVHN	18.9(4.9) /35.3(2.9) /43.3(2.7) /52.0(0.6) /67.1(3.4) /77.0(5.0)
	Uniform	2.95(4.1) /66.1(46.) /100.(0.0) /0.0(0.0) /100.(0.0) /100.(0.0)
	Gaussian	0.06(0.0) /33.3(47.) /100.(0.0) /0.0(0.0) /100.(0.0) /100.(0.0)
CIFAR-10	Imagenet(c)	50.0(2.8) /47.8(15.) /81.2(0.8) /92.0(0.2) /90.1(1.5) /93.4(1.2)
	Imagenet(r)	47.4(4.4) /51.9(16.) /90.9(0.5) /93.6(0.2) /91.7(1.6) /95.8(0.9)
	LSUN(c)	51.8(3.1) /63.5(7.8) /64.2(0.6) /92.5(0.4) /93.3(1.5) /91.5(1.2)
	LSUN(r)	56.3(3.6) /59.2(18.) /91.7(0.3) /94.9(0.2) /95.7(0.1) /97.6(0.5)
	iSUN	52.3(3.6) /57.2(18.) /90.6(0.7) /94.6(0.3) /95.4(0.2) /97.5(0.3)
	SVHN	40.5(6.9) /48.7(3.2) /90.6(1.7) /91.4(1.1) /92.1(3.4) /94.0(0.6)
	Uniform	59.9(12.) /98.1(2.6) /100.(0.0) /99.9(0.0) /100.(0.0) /100.(0.0)
	Gaussian	48.8(26.) /92.1(11.) /100.(0.0) /99.9(0.0) /100.(0.0) /100.(0.0)

Table A.5: Performance of six OOD detection methods on 8 benchmark datasets. The experiment here is the same as Table 7.1 but use Resnet-34 for the backbone network. The value in parentheses is the standard deviation.

ID	OOD	AUROC
		Baseline / ODIN* / Maha* / DeConf-I* / DeConf-E* / DeConf-C*
CIFAR-100	Imagenet(c)	78.9(0.1) /84.8(0.6) /93.4(0.3) /88.2(0.6) /95.2(0.6) /95.3(0.6)
	Imagenet(r)	75.1(0.8) /85.7(0.2) /96.3(0.1) /84.6(1.0) /97.0(0.4) /95.9(0.7)
	LSUN(c)	78.8(0.6) /80.3(1.3) /79.8(0.3) /93.8(0.3) /92.6(0.2) /93.8(0.3)
	LSUN(r)	76.2(1.4) /86.6(0.8) /96.3(0.2) /85.9(1.8) /97.0(0.7) /96.1(0.5)
	iSUN	75.2(1.4) /85.9(0.8) /95.8(0.2) /84.7(1.4) /96.6(0.6) /95.7(0.5)
	SVHN	75.1(2.5) /80.2(2.0) /80.9(1.1) /89.2(2.6) /93.8(0.8) /93.2(1.1)
	Uniform	69.0(13.) /96.7(2.5) /100.(0.0) /79.3(8.3) /99.9(0.0) /99.9(0.0)
	Gaussian	51.5(1.8) /93.7(1.7) /99.9(0.0) /60.8(23.) /99.9(0.0) /99.9(0.0)
CIFAR-10	Imagenet(c)	90.0(0.9) /81.2(2.4) /94.2(0.1) /98.2(0.2) /98.2(0.1) /96.0(0.2)
	Imagenet(r)	87.3(1.3) /81.1(2.9) /96.5(0.1) /98.1(0.3) /98.1(0.3) /96.1(0.5)
	LSUN(c)	92.0(1.7) /77.9(4.6) /87.7(0.2) /98.8(0.1) /98.5(0.0) /97.2(0.1)
	LSUN(r)	91.6(1.2) /88.5(2.0) /97.2(0.1) /98.9(0.2) /99.0(0.1) /98.0(0.1)
	iSUN	90.1(1.4) /86.1(2.5) /96.5(0.2) /98.8(0.2) /98.9(0.1) /97.6(0.1)
	SVHN	87.7(2.4) /63.9(4.3) /87.8(1.6) /96.8(0.4) /96.1(1.4) /97.8(0.3)
	Uniform	85.9(10.) /93.3(4.5) /99.9(0.0) /99.6(0.1) /99.9(0.0) /99.9(0.0)
	Gaussian	89.9(10.) /97.1(2.0) /99.9(0.0) /99.7(0.0) /99.9(0.0) /99.9(0.0)
ID	OOD	TNR@TPR95
		Baseline / ODIN* / Maha* / DeConf-I* / DeConf-E* / DeConf-C*
CIFAR-100	Imagenet(c)	24.1(0.6) /44.0(2.2) /68.2(1.4) /42.6(2.7) /73.4(3.7) /72.6(3.7)
	Imagenet(r)	19.4(0.1) /45.5(1.4) /82.6(0.8) /30.4(3.0) /84.3(2.7) /76.5(3.8)
	LSUN(c)	21.9(0.4) /34.8(2.4) /27.7(1.4) /66.1(2.2) /59.7(0.7) /65.7(2.3)
	LSUN(r)	19.8(1.6) /48.2(3.0) /81.8(1.4) /29.4(5.2) /84.6(4.0) /76.8(3.3)
	iSUN	17.7(0.5) /45.3(2.8) /80.4(0.8) /27.1(4.3) /83.0(3.1) /75.3(3.3)
	SVHN	16.6(1.5) /27.5(5.0) /25.7(2.6) /43.7(10.) /60.8(5.3) /55.1(7.1)
	Uniform	5.63(7.0) /76.4(27.) /100.(0.0) /4.11(5.8) /100.(0.0) /100.(0.0)
	Gaussian	0.0(0.0) /46.6(20.) /100.(0.0) /0.06(0.0) /100.(0.0) /100.(0.0)
CIFAR-10	Imagenet(c)	54.6(2.6) /53.7(3.1) /74.6(0.6) /90.8(1.5) /91.1(0.9) /81.1(1.7)
	Imagenet(r)	48.3(3.2) /53.1(4.3) /85.1(0.6) /90.5(1.8) /90.8(1.8) /81.4(2.4)
	LSUN(c)	59.9(4.7) /50.9(6.1) /53.6(1.0) /93.9(0.5) /92.4(0.5) /87.3(1.0)
	LSUN(r)	57.5(4.4) /68.1(4.2) /87.4(0.8) /95.8(1.0) /96.0(0.7) /90.9(0.9)
	iSUN	53.7(3.8) /62.8(5.0) /84.6(0.9) /95.1(1.0) /95.3(0.5) /88.8(1.1)
	SVHN	44.5(8.1) /29.7(6.2) /46.2(4.8) /84.5(2.5) /78.8(7.6) /89.5(2.1)
	Uniform	27.9(20.) /74.5(20.) /100.(0.0) /100.(0.0) /100.(0.0) /100.(0.0)
	Gaussian	52.7(40.) /87.1(9.3) /100.(0.0) /100.(0.0) /100.(0.0) /100.(0.0)

Table A.6: The AUROC of individual experimental setting in Figures 7.4 and 7.5. The experiments do not use input preprocessing. All values are percentages averaged over three runs, and the value in parentheses is the standard deviation. The "+" means that the classifier is trained with extra regularization (dropout rate 0.7).

ID	OoD	Plain-I	DeConf-I-$h(x)$	DeConf-I-$g(x)$	Plain-E	DeConf-E-$h(x)$	DeConf-E-$g(x)$	Plain-C	DeConf-C-$h(x)$	DeConf-C-$g(x)$
SVHN	Imagenet(c)	92.8(1.0)	98.7(0.1)	98.4(0.0)	92.9(0.8)	96.2(0.4)	97.2(0.7)	60.3(1.4)	93.9(0.4)	91.3(6.0)
	Imagenet(r)	92.4(0.8)	98.7(0.1)	98.4(0.0)	93.0(0.6)	96.1(0.4)	97.0(0.9)	63.9(1.5)	93.5(0.7)	90.8(6.0)
	LSUN(c)	91.0(0.6)	98.0(0.2)	97.5(0.5)	92.0(0.6)	95.0(0.1)	95.5(0.4)	51.9(0.3)	93.1(0.3)	92.4(5.8)
	LSUN(r)	91.3(1.0)	98.4(0.2)	98.3(0.3)	92.1(0.7)	95.5(0.7)	96.7(1.1)	61.4(1.6)	92.0(0.7)	90.6(5.8)
	iSUN	91.5(0.9)	98.6(0.1)	98.5(0.2)	92.4(0.8)	95.8(0.6)	96.7(1.1)	60.9(1.6)	93.1(0.8)	91.3(5.8)
	CIFAR10	91.4(0.7)	98.4(0.1)	98.0(0.0)	92.6(0.5)	95.6(0.4)	96.9(0.5)	63.1(1.7)	93.3(0.6)	89.5(5.8)
	CIFAR100	91.3(0.4)	98.1(0.1)	97.3(0.1)	92.5(0.5)	95.0(0.5)	96.3(0.6)	64.0(1.6)	93.0(0.5)	88.5(6.1)
	Uniform	93.7(1.6)	98.7(0.2)	98.6(0.3)	93.0(0.7)	94.8(0.9)	93.9(1.8)	64.6(2.4)	95.1(1.6)	90.1(6.4)
	Gaussian	94.4(1.3)	98.8(0.2)	98.7(0.3)	93.6(0.3)	95.6(0.6)	95.4(1.6)	66.5(3.4)	95.1(1.4)	90.3(6.5)
CIFAR-10	Imagenet(c)	90.0(0.9)	97.7(0.4)	96.6(0.7)	92.3(0.4)	97.4(0.2)	96.6(0.7)	74.3(2.9)	96.4(0.3)	87.3(10.)
	Imagenet(r)	87.3(1.3)	96.9(0.6)	95.5(1.0)	91.2(0.3)	96.8(0.4)	95.4(1.4)	71.7(3.1)	95.6(0.5)	86.0(12.)
	LSUN(c)	92.0(1.7)	99.0(0.0)	98.8(0.0)	93.7(0.5)	98.7(0.0)	98.7(0.1)	79.9(3.1)	98.4(0.0)	92.0(5.7)
	LSUN(r)	91.6(1.2)	98.2(0.4)	96.1(1.2)	93.5(0.3)	98.1(0.1)	95.8(1.3)	72.8(2.9)	97.6(0.2)	85.2(14.)
	iSUN	90.1(1.4)	98.0(0.4)	96.2(1.1)	92.9(0.4)	97.9(0.1)	96.0(1.2)	71.0(3.7)	97.3(0.3)	86.0(13.)
	SVHN	87.7(2.4)	98.3(0.4)	99.3(0.3)	91.7(0.7)	97.5(0.8)	99.0(0.3)	80.6(5.1)	98.6(0.5)	92.5(4.3)
	Uniform	85.9(10.)	93.5(1.0)	97.7(1.6)	88.6(2.5)	99.2(0.6)	93.3(9.0)	75.1(14.)	99.6(0.1)	87.2(5.2)
	Gaussian	89.9(10.)	94.6(1.4)	98.3(0.9)	89.6(5.1)	99.2(0.5)	88.6(15.)	77.8(3.4)	99.6(0.2)	85.1(1.5)
CIFAR-100	Imagenet(c)	78.9(0.1)	83.2(0.6)	63.7(6.6)	76.0(1.5)	93.4(0.7)	57.0(11.)	64.6(0.3)	92.6(0.8)	46.3(10.)
	Imagenet(r)	75.1(0.8)	76.6(1.4)	50.1(9.1)	72.0(1.8)	95.5(0.6)	46.6(15.)	61.8(1.5)	91.8(1.1)	51.3(14.)
	LSUN(c)	78.8(0.6)	91.3(0.5)	85.7(1.4)	77.5(0.3)	90.1(0.6)	78.1(3.0)	60.5(1.0)	93.3(0.7)	35.6(1.4)
	LSUN(r)	76.2(1.4)	78.4(2.5)	46.0(10.)	71.3(0.7)	95.5(0.8)	43.0(14.)	64.1(1.8)	92.0(0.7)	44.6(12.)
	iSUN	75.2(1.4)	76.6(2.0)	45.7(10.)	71.4(1.1)	95.2(0.7)	40.0(15.)	61.9(1.9)	91.6(0.8)	45.0(14.)
	SVHN	75.1(2.5)	89.6(2.0)	87.9(3.3)	77.5(2.2)	91.5(1.7)	75.3(7.8)	60.0(5.2)	93.6(1.3)	54.3(4.8)
	Uniform	69.0(13.)	50.4(11.)	46.8(26.)	84.0(10.)	99.8(0.1)	25.0(17.)	59.2(36.)	99.6(0.1)	97.7(0.8)
	Gaussian	51.5(1.8)	31.9(17.)	27.0(19.)	84.8(5.7)	99.9(0.0)	7.75(4.2)	23.5(14.)	99.3(0.3)	99.4(0.4)
CIFAR-100+	Imagenet(c)	77.0(1.4)	87.0(0.1)	82.1(1.2)	78.2(0.4)	86.8(1.2)	83.1(1.8)	69.4(3.4)	88.3(1.1)	81.6(0.7)
	Imagenet(r)	73.7(1.4)	83.8(0.6)	76.5(2.6)	76.3(0.5)	84.3(1.5)	78.0(2.7)	72.4(2.8)	87.0(1.1)	75.4(1.2)
	LSUN(c)	77.6(0.5)	89.3(0.5)	89.8(0.8)	76.5(1.1)	90.0(0.3)	90.7(0.7)	55.0(2.3)	86.2(1.3)	89.6(1.1)
	LSUN(r)	75.4(2.6)	84.6(1.3)	76.8(1.9)	75.8(2.0)	84.5(0.4)	78.0(2.8)	70.1(3.8)	87.0(2.2)	74.8(2.2)
	iSUN	74.5(1.6)	83.6(0.8)	76.2(2.4)	74.9(1.1)	84.0(1.1)	77.7(2.0)	67.9(3.8)	85.7(1.7)	74.5(1.8)
	SVHN	72.2(5.9)	83.2(2.9)	81.4(5.2)	74.5(3.4)	86.0(2.6)	83.8(3.4)	67.8(2.4)	86.3(2.7)	84.9(2.2)
	Uniform	87.0(1.5)	85.1(6.1)	75.6(14.)	88.7(3.2)	95.0(2.1)	83.6(11.)	60.0(15.)	81.7(14.)	57.3(2.4)
	Gaussian	87.1(4.7)	84.2(10.)	81.6(14.)	77.7(4.3)	95.4(4.7)	85.1(10.)	70.0(2.2)	75.9(4.6)	34.9(4.6)

REFERENCES

[1] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei, "Whats the point: Semantic segmentation with point supervision," in *European Conference on Computer Vision*, Springer, 2016, pp. 549–565.

[2] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 132–149.

[3] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5147–5156.

[4] G. Larsson, M. Maire, and G. Shakhnarovich, "Colorization as a proxy task for visual understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6874–6883.

[5] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European conference on computer vision*, Springer, 2016, pp. 649–666.

[6] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," *arXiv preprint arXiv:1803.07728*, 2018.

[7] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1422–1430.

[8] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *European Conference on Computer Vision*, Springer, 2016, pp. 69–84.

[9] Y.-C. Hsu and Z. Kira, "Neural network-based clustering using pairwise constraints," in *ICLR workshop*, 2016.

[10] Y.-C. Hsu, Z. Lv, J. Schlosser, and Z. K. Phillip Odom, "Multi-class classification without multi-class labels," in *International Conference on Learning Representations (ICLR)*, 2019.

[11] Y.-C. Hsu, Z. Lv, and Z. Kira, "Learning to cluster in order to transfer across domains and tasks," in *International Conference on Learning Representations (ICLR)*, 2018.

[12] H. J.Z. K. Yen-Chang Hsu Yilin Shen, "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2020.

[13] Y.-C. Hsu, Z. Xu, Z. Kira, and J. Huang, "Learning to cluster for proposal-free instance segmentation," in *International Joint Conference on Neural Networks (IJCNN)*, 2018.

[14] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Advances in neural information processing systems*, 2003, pp. 521–528.

[15] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th international conference on Machine learning*, ACM, 2007, pp. 209–216.

[16] S. Anand, S. Mittal, O. Tuzel, and P. Meer, "Semi-supervised kernel mean shift clustering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 6, pp. 1201–1215, 2014.

[17] E. Amid, A. Gionis, and A. Ukkonen, "Semi-supervised kernel metric learning using relative comparisons," *arXiv preprint arXiv:1612.00086*, 2016.

[18] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, *et al.*, "Constrained k-means clustering with background knowledge," in *ICML*, vol. 1, 2001, pp. 577–584.

[19] X. Wang, B. Qian, and I. Davidson, "On constrained spectral clustering and its applications," *Data Mining and Knowledge Discovery*, pp. 1–30, 2014.

[20] S. S. Rangapuram and M. Hein, "Constrained 1-spectral clustering.," in *AISTATS*, vol. 30, 2012, p. 90.

[21] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering," in *Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004, p. 11.

[22] V. Antoine, B. Quost, M.-H. Masson, and T. Denoeux, "Cecm: Constrained evidential c-means algorithm," *Computational Statistics & Data Analysis*, vol. 56, no. 4, pp. 894–914, 2012.

[23] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA., vol. 1, 1967, pp. 281–297.

[24] X. Chen and D. Cai, "Large scale spectral clustering with landmark-based representation.," in *AAAI*, vol. 5, 2011, p. 14.

[25] D. Cai, X. He, X. Wang, H. Bao, and J. Han, "Locality preserving nonnegative matrix factorization.," in *IJCAI*, vol. 9, 2009, pp. 1010–1015.

[26] I. Davidson and S. Basu, "A survey of clustering with instance level," *Constraints*, vol. 1, p. 2, 2007.

[27] D. Dinler and M. K. Tural, "A survey of constrained clustering," in *Unsupervised Learning Algorithms*, Springer, 2016, pp. 207–235.

[28] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. on Knowl. and Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[29] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition.," in *ICML*, 2014.

[30] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" In *NIPS*, 2014.

[31] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition*, 2014.

[32] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[33] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[34] J. Hoffman, E. Rodner, J. Donahue, K. Saenko, and T. Darrell, "Efficient learning of domain-invariant image representations," in *International Conference on Learning Representations (ICLR)*, 2013.

[35] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[36] E. Tzeng*, J. Hoffman*, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *International Conference in Computer Vision (ICCV)*, 2015.

[37] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, pp. 97–105.

[38] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016.

[39] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation.," in *AAAI*, 2016.

[40] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," in *Advances in Neural Information Processing Systems*, 2016.

[41] O. Sener, H. O. Song, A. Saxena, and S. Savarese, "Learning transferrable representations for unsupervised domain adaptation," in *Advances in Neural Information Processing Systems*, 2016, pp. 2110–2118.

[42] F. M. Carlucci, L. Porzi, B. Caputo, E. Ricci, and S. Rota Bulò, "Autodial: Automatic domain alignment layers," in *International Conference on Computer Vision*, 2017.

[43] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 2208–2217.

[44] W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz, "Central moment discrepancy (cmd) for domain-invariant representation learning," *ICLR*, 2017.

[45] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[46] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Adversarial discriminative domain adaptation," in *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[47] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 4, pp. 801–814, 2018.

[48] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle consistent adversarial domain adaptation," in *International Conference in Machine Learning (ICML)*, 2018.

[49] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann, "Contrastive adaptation network for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4893–4902.

[50] Y. Luo, L. Zheng, T. Guan, J. Yu, and Y. Yang, "Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2507–2516.

[51] Y. Zhang, H. Tang, K. Jia, and M. Tan, "Domain-symmetric networks for adversarial domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5031–5040.

[52] M. Rohrbach, S. Ebert, and B. Schiele, "Transfer learning in a transductive setting," in *Advances in neural information processing systems*, 2013, pp. 46–54.

[53] W. Ge and Y. Yu, "Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1086–1095.

[54] Y. Cui, Y. Song, C. Sun, A. Howard, and S. Belongie, "Large scale fine-grained categorization and domain-specific transfer learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4109–4118.

[55] P. Panareda Busto and J. Gall, "Open set domain adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 754–763.

[56] K. Saito, S. Yamamoto, Y. Ushiku, and T. Harada, "Open set domain adaptation by backpropagation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 153–168.

[57] K. You, M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Universal domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2720–2729.

[58] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *International Conference on Learning Representations (ICLR)*, 2017.

[59] S. Liang, Y. Li, and R Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *arXiv preprint arXiv:1706.02690*, 2017.

[60] A. Bendale and T. Boult, "Towards open world recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1893–1902.

[61] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1563–1572.

[62] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012.

[63] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CVPR*, 2015.

[64] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1529–1537.

[65] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *arXiv preprint arXiv:1504.00702*, 2015.

[66] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1764–1772.

[67] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[68] C. Clark and A. Storkey, "Training deep convolutional neural networks to play go," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 1766–1774.

[69] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.

[70] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering.," in *AAAI*, 2014, pp. 1293–1299.

[71] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *2014 22nd International Conference on Pattern Recognition (ICPR)*, IEEE, 2014, pp. 1532–1537.

[72] M. Shao, S. Li, Z. Ding, and Y. Fu, "Deep linear coding for fast graph clustering," in *Proceedings of the 24th International Conference on Artificial Intelligence*, AAAI Press, 2015, pp. 3798–3804.

[73] Z. Wang, S. Chang, J. Zhou, and T. S. Huang, "Learning a task-specific deep architecture for clustering," *Proceedings of SIAM Conference on Data Mining (SDM)*, 2016.

[74] G. Chen, "Deep learning with nonparametric clustering," *arXiv preprint arXiv:1501.03084*, 2015.

[75] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, Springer, 2013, pp. 117–124.

[76] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," *arXiv preprint arXiv:1511.06335*, 2015.

[77] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev, "Metric learning with adaptive density discrimination," *arXiv preprint arXiv:1511.05939*, 2015.

[78] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using a siamesei time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 04, pp. 669–688, 1993.

[79] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, IEEE, vol. 1, 2005, pp. 539–546.

[80] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, IEEE, vol. 2, 2006, pp. 1735–1742.

[81] J. Weston, F. Ratle, and R. Collobert, "Deep learning via semi-supervised embedding," in *Proceedings of the 25th International Conference on Machine Learning (ICML-08)*, 2008, pp. 1168–1175.

[82] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "Matchnet: Unifying feature and metric learning for patch-based matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3279–3286.

[83] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[84] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proceedings of the International Conference on Computer Vision*, 2015.

[85] H. Mobahi, R. Collobert, and J. Weston, "Deep learning from temporal coherence in video," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, 2009, pp. 737–744.

[86] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun, "Unsupervised feature learning from temporal data," *arXiv preprint arXiv:1504.02518*, 2015.

[87] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, IEEE, 2014, pp. 1386–1393.

[88] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[89] S. S. Rangapuram and M. Hein, "Constrained 1-spectral clustering," *International conference on Artificial Intelligence and Statistics*, 2012.

[90] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering," in *Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004, p. 11.

[91] R. C. De Amorim and B. Mirkin, "Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering," *Pattern Recognition*, vol. 45, no. 3, pp. 1061–1075, 2012.

[92] D. Khashabi, J. Y. Liu, J. Wieting, and F. Liang, "Clustering with side information: From a probabilistic model to a deterministic algorithm," *arXiv preprint arXiv:1508.06235*, 2015.

[93] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison,

A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019.

[94] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[95] A. Krizhevsky and G. Hinton, *Learning multiple layers of features from tiny images*, 2009.

[96] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[97] A. Strehl and J. Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *The Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2003.

[98] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.

[99] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[100] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[101] W. Dai, Y. Chen, G. rong Xue, Q. Yang, and Y. Yu, "Translated learning: Transfer learning across different feature spaces," in *Advances in Neural Information Processing Systems 21*, 2009.

[102] K. Han, A. Vedaldi, and A. Zisserman, "Learning to discover novel visual categories via deep transfer clustering," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8401–8409.

[103] K. Han, S.-A. Rebuffi, S. Ehrhardt, A. Vedaldi, and A. Zisserman, "Automatically discovering and learning new visual categories with ranking statistics," in *ICLR*, 2020.

[104] G. Huang, H. Larochelle, and S. Lacoste-Julien, "Centroid networks for few-shot clustering and unsupervised few-shot classification," *arXiv preprint arXiv:1902.08605*, 2019.

[105] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4353–4361.

[106] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[107] A. Strehl and J. Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *Journal of machine learning research*, vol. 3, no. Dec, pp. 583–617, 2002.

[108] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang, "Image clustering using local discriminant models and global integration," *IEEE Transactions on Image Processing*, 2010.

[109] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," *NIPS*, 2016.

[110] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, 2015.

[111] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Computer Vision – ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 213–226.

[112] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[113] K. Saito, Y. Ushiku, and T. Harada, "Asymmetric tri-training for unsupervised domain adaptation," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 2988–2997.

[114] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[115] Y. LeCun, "The mnist database of handwritten digits," 1998.

[116] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *Journal of machine learning research*, vol. 1, no. Dec, pp. 113–141, 2000.

[117] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognition*, vol. 44, no. 8, pp. 1761–1776, 2011.

[118] R. Anand, K. Mehrotra, C. K. Mohan, and S. Ranka, "Efficient classification for multiclass problems using modular neural networks," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 117–124, 1995.

[119] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *Journal of machine learning research*, vol. 5, no. Jan, pp. 101–141, 2004.

[120] S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: A stepwise procedure for building and training a neural network," in *Neurocomputing*, Springer, 1990, pp. 41–50.

[121] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in *Advances in neural information processing systems*, 1998, pp. 507–513.

[122] T.-F. Wu, C.-J. Lin, and R. C. Weng, "Probability estimates for multi-class classification by pairwise coupling," *Journal of Machine Learning Research*, vol. 5, no. Aug, pp. 975–1005, 2004.

[123] J. Weston and C. Watkins, "Multi-class support vector machines," Citeseer, Tech. Rep., 1998.

[124] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[125] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine learning*, vol. 37, no. 3, pp. 297–336, 1999.

[126] J. Fürnkranz, "Round robin ensembles," *Intelligent Data Analysis*, vol. 7, no. 5, pp. 385–403, 2003.

[127] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *International Conference on Learning Representations*, 2017.

[128] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 1163–1171.

[129] T. Miyato, S.-i. Maeda, S. Ishii, and M. Koyama, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[130] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Advances in neural information processing systems*, 2017, pp. 1195–1204.

[131] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," 2013.

[132] H. Bao, G. Niu, and M. Sugiyama, "Classification from pairwise similarity and unlabeled data," in *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[133] Y.-C. Hsu and Z. Kira, "Neural network-based clustering using pairwise constraints," *International Conference on Learning Representations (ICLR) workshop*, 2016.

[134] Y.-C. Hsu, Z. Lv, and Z. Kira, "Learning to cluster in order to transfer across domains and tasks," in *International Conference on Learning Representations (ICLR)*, 2018.

[135] H. Li, Z. Xu, G. Taylor, and T. Goldstein, "Visualizing the loss landscape of neural nets," *arXiv preprint arXiv:1712.09913*, 2017.

[136] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[137] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[138] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[139] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision*, Springer, 2016, pp. 630–645.

[140] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[141] I. J. Goodfellow, O. Vinyals, and A. M. Saxe, "Qualitatively characterizing neural network optimization problems," *arXiv preprint arXiv:1412.6544*, 2014.

[142] D. J. Im, M. Tao, and K. Branson, "An empirical analysis of deep network loss surfaces," *arXiv preprint arXiv:1612.04010*, 2016.

[143] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR*, 2009.

[144] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," *arXiv preprint arXiv:1804.09170*, 2018.

[145] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" In *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[146] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

[147] S. Rabanser, S. Günnemann, and Z. Lipton, "Failing loudly: An empirical study of methods for detecting dataset shift," in *Advances in Neural Information Processing Systems*, 2019, pp. 1394–1406.

[148] W. J. Conover and W. J. Conover, "Practical nonparametric statistics," 1980.

[149] L. N. Wasserstein, "Markov processes with countable state space describing large systems of automata," 1969.

[150] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. Mar, pp. 723–773, 2012.

[151] A. Ramdas, N. G. Trillos, and M. Cuturi, "On wasserstein two-sample testing and related families of nonparametric tests," *Entropy*, vol. 19, no. 2, p. 47, 2017.

[152] B. Piccoli and F. Rossi, "Generalized wasserstein distance and its application to transport equations with source," *Archive for Rational Mechanics and Analysis*, vol. 211, no. 1, pp. 335–358, 2014.

[153] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *European conference on computer vision*, Springer, 2016, pp. 443–450.

[154] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," *arXiv preprint arXiv:1812.01754*, 2018.

[155] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. Lanckriet, "Hilbert space embeddings and metrics on probability measures," *Journal of Machine Learning Research*, vol. 11, no. Apr, pp. 1517–1561, 2010.

[156] M. E. Hellman, "The nearest neighbor classification rule with a reject option," *IEEE Transactions on Systems Science and Cybernetics*, vol. 6, no. 3, pp. 179–185, 1970.

[157] Y. Geifman and R. El-Yaniv, "Selective classification for deep neural networks," in *Advances in neural information processing systems*, 2017, pp. 4878–4887.

[158] C. Cortes, G. DeSalvo, and M. Mohri, "Learning with rejection," in *International Conference on Algorithmic Learning Theory*, Springer, 2016, pp. 67–82.

[159] J Andrews, T. Tanay, E. J. Morton, and L. D. Griffin, "Transfer representation-learning for anomaly detection," in *JMLR*, 2016.

[160] A. Malinin and M. Gales, "Predictive uncertainty estimation via prior networks," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018, pp. 7047–7058.

[161] ——, "Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 520–14 531.

[162] A. Malinin, B. Mlodozeniec, and M. Gales, "Ensemble distribution distillation," in *International Conference on Learning Representations*, 2019.

[163] A. Shafaei, M. Schmidt, and J. Little, "A less biased evaluation of ood sample detectors," in *Proceedings of the British Machine Vision Conference (BMVC)*, BMVA Press, 2019.

[164] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[165] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, 2017, pp. 6402–6413.

[166] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," *International Conference on Learning Representations (ICLR)*, 2017.

[167] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems*, 2018, pp. 7167–7177.

[168] A. Vyas, N. Jammalamadaka, X. Zhu, D. Das, B. Kaul, and T. L. Willke, "Out-of-distribution detection using an ensemble of self supervised leave-out classifiers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 550–564.

[169] A. R. Dhamija, M. Günther, and T. Boult, "Reducing network agnostophobia," 2018.

[170] D. Hendrycks, M. Mazeika, and T. G. Dietterich, "Deep anomaly detection with outlier exposure," *International Conference on Learning Representations (ICLR)*, 2019.

[171] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," *International Conference on Learning Representations (ICLR)*, 2018.

[172] G. Shalev, Y. Adi, and J. Keshet, "Out-of-distribution detection using multiple semantic label representations," in *Advances in Neural Information Processing Systems*, 2018, pp. 7375–7385.

[173] M. Masana, I. Ruiz, J. Serrat, J. van de Weijer, and A. M. Lopez, "Metric learning for novelty and anomaly detection," *arXiv preprint arXiv:1808.05492*, 2018.

[174] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, "Visual domain adaptation: A survey of recent advances," *IEEE signal processing magazine*, vol. 32, no. 3, pp. 53–69, 2015.

[175] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[176] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 427–436.

[177] L. Neumann, A. Zisserman, and A. Vedaldi, "Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection," 2018.

[178] E. Techapanurak and T. Okatani, "Hyperparameter-free out-of-distribution detection using softmax of scaled cosine similarity," *CoRR*, vol. abs/1905.10628, 2019.

[179] X. Zhang, R. Zhao, Y. Qiao, X. Wang, and H. Li, "Adacos: Adaptively scaling cosine logits for effectively learning deep face representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 823–10 832.

[180] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[181] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 1321–1330.

[182] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. A. DePristo, J. V. Dillon, and B. Lakshminarayanan, "Likelihood ratios for out-of-distribution detection," *arXiv preprint arXiv:1906.02845*, 2019.

[183] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?" *arXiv preprint arXiv:1810.09136*, 2018.

[184] H. Choi and E. Jang, "Generative ensembles for robust anomaly detection," *arXiv preprint arXiv:1810.01392*, 2018.

[185] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[186] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

[187] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[188] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[189] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," *CVPR*, 2018.

[190] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[191] Z. Hayder, X. He, and M. Salzmann, "Boundary-aware instance segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[192] S. Liu, J. Jia, S. Fidler, and R. Urtasun, "Sgn: Sequential grouping networks for instance segmentation," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[193] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[194] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother, "Instancecut: From edges to instances with multicut," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[195] B. D. Brabandere, D. Neven, and L. V. Gool, "Semantic instance segmentation with a discriminative loss function," *CoRR*, vol. abs/1708.02551, 2017.

[196] Y. Liu, S. Yang, B. Li, W. Zhou, J. Xu, H. Li, and Y. Lu, "Affinity derivation and graph merge for instance segmentation," in *European Conference on Computer Vision*, Springer, 2018, pp. 708–724.

[197] E. Levinkov, J. Uhrig, S. Tang, M. Omran, E. Insafutdinov, A. Kirillov, C. Rother, T. Brox, B. Schiele, and B. Andres, "Joint graph decomposition & node labeling: Problem, algorithms, applications," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[198] J. D.X. J. Yi Li Haozhi Qi and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[199] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3150–3158.

[200] S. Zagoruyko, A. Lerer, T. Lin, P. O. Pinheiro, S. Gross, S. Chintala, and P. Dollár, "A multipath network for object detection," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

[201] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016, pp. 379–387.

[202] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[203] B. Romera-Paredes and P. H. S. Torr, "Recurrent instance segmentation," in *European Conference on Computer Vision*, Springer, 2016, pp. 312–329.

[204] A. Arnab and P. H. S. Torr, "Pixelwise instance segmentation with a dynamically instantiated network," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[205] S. Chopra and M. R. Rao, "The partition problem," *Mathematical Programming*, 1993.

[206] X. Liang, Y. Wei, X. Shen, J. Yang, L. Lin, and S. Yan, "Proposal-free network for instance-level object segmentation," *arXiv preprint arXiv:1509.02636*, 2015.

[207] J. Uhrig, M. Cordts, U. Franke, and T. Brox, "Pixel-level encoding and depth layering for instance-level semantic labeling," in *German Conference on Pattern Recognition*, Springer, 2016, pp. 14–25.

[208] J. Dai, K. He, Y. Li, S. Ren, and J. Sun, "Instance-sensitive fully convolutional networks," in *European Conference on Computer Vision*, Springer, 2016, pp. 534–549.

[209] L. Jin, Z. Chen, and Z. Tu, "Object detection free instance segmentation with labeling transformations," *arXiv preprint arXiv:1611.08991*, 2016.

[210] K. Appel and W. Haken, "The four-color problem," *Mathematics today: Twelve informal essays*, pp. 153–180, 1978.

[211] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.

[212] *Cvpr 2017 workshop on autonomous driving challenge*, 2017 (accessed November 23, 2017).

[213] *Cvpr 2017 workshop on autonomous driving challenge leaderboard*, 2017 (accessed July 21, 2017).

[214] P. L.X. W. Xingang Pan Jianping Shi and X. Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[215] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[216] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, "Large scale online learning of image similarity through ranking," *Journal of Machine Learning Research*, vol. 11, no. Mar, pp. 1109–1135, 2010.

[217] W. Chen, X. Chen, J. Zhang, and K. Huang, "Beyond triplet loss: A deep quadruplet network for person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 403–412.

[218] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4004–4012.

[219] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," in *Neural Networks*, Elsevier, 2019.

[220] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, and Z. Kira, "Re-evaluating continual learning scenarios: A categorization and case for strong baselines," in *NeurIPS Continual learning Workshop*, 2018.